Universidad Carlos iii de Madrid

Escuela Politécnica Superior

Ingeniería Industrial

PROYECTO FIN DE CARRERA

ESTIMACIÓN DE LA DISTANCIA RECORRIDA POR UN ROBOT MÓVIL MEDIANTE LA UTILIZACIÓN DE DESCRIPTORES SURF

Autor: Juan Manuel Peraza Domínguez
Tutor: Dr. Luis Moreno Lorente

Noviembre de 2009

A mi abuelo Rafael, un modelo en el que fijarse.

Agradecimientos

A mi tutor: D. Luis Moreno Lorente, por darme la oportunidad de realizar el proyecto bajo su tutela y sin cuyo valioso tiempo, conocimientos y ayuda no hubiera sido posible sacarlo adelante.

A mi amigo Fernando Martín Monar, que no ha dudado en ningún momento en ofrecerme su tiempo y ayuda, convirtiéndose en un apoyo importantísimo para mí.

A mis compañeros de laboratorio: César, Fernando, Lucas, Fran..., sin cuya compañía las horas de trabajo no hubieran sido tan agradables y por no dudar a la hora de resolverme cualquier duda.

Al Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III, por poner al servicio todos los medios y conocimientos posibles para la realización del proyecto.

A Arturo Gil, de la Universidad Miguel Hernández de Elche, por no vacilar a la hora de prestar sus conocimientos.

A mi familia y a los que aunque no lo sean los considero como tal, que me lo han dado todo y no han dejado de creer ni un segundo en mí.

A mis amigos, que siempre han estado ahí.

A Ana, la persona más especial que ha entrado en mi vida y para la que no hay palabras suficientes de agradecimiento.

A todos los profesores que de alguna manera han contribuido en mi formación, y, de manera especial, aquellos que me han hecho disfrutar de lo aprendido.

En definitiva, a todos aquellos que no han sido nombrados y que de cualquier manera, directa o indirecta, han tenido que ver con la realización de este proyecto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

A lo largo de la historia de la humanidad, el ser humano ha tratado de desarrollar todo tipo de máquinas que de algún modo u otro buscaban automatizar procesos de la vida cotidiana del hombre, de manera que estos artefactos hiciesen más fácil la vida de éste. Con el paso de los años estas máquinas han ido ganando en complejidad y recibiendo todo tipo de nombres, hasta que en el año 1921, de la mano del escritor checo Karel Capek, fue usado, por primera vez, el término robot en una novela de ficción. Lo que parecía fruto de la imaginación del hombre o más propio de historias de ciencia ficción poco a poco ha ido transformándose en realidad. A partir de la segunda mitad del siglo XX, comienzan a producirse numerosísimos avances dentro del mundo de la Robótica. Lo que en un principio parecía tener una mera aplicación industrial, ha ido abriéndose paso de manera vertiginosa en la vida del hombre, llegando a tener aplicación en cada vez más numerosos y diferentes ámbitos, tales como la exploración espacial, la medicina, el armamento militar, y un largo etcétera.

Uno de los campos de estudio dentro de la Robótica es la navegación autónoma de los robots. Desde hace muchos años, numerosos investigadores trabajan día a día para conseguir que los robots sean cada vez más autónomos

y puedan desplazarse por entornos cada vez más complejos con una mayor facilidad e independencia. Completamente relacionado con este ámbito de la Robótica se encuentra la Visión por Computador, cuya evolución y aplicación han estado estrechamente ligadas al desarrollo y mejora de la informática a partir de los años 80. La Visión por Computador ha sido uno de los campos que ha experimentado un mayor y más rápido progreso en los últimos años, siendo aplicada para sistemas de control de calidad o de navegación robótica, ámbito en el cual tiene cabida el presente proyecto. Para que el robot tome conciencia del entorno que le rodea, tenemos a nuestra disposición diferentes tipos de sensores que pueden proporcionarnos una imagen del mismo, como son los rayos X, infrarrojos, ultrasonidos o el empleo de cámaras digitales de video, caso en el que nos vamos a centrar.

Por tanto, el objetivo que persigue el Proyecto Final de Carrera descrito en este documento es el desarrollo de un algoritmo para la obtención de la posición, centrándonos en el cálculo de la distancia recorrida y la velocidad de un robot móvil que desempeñe su actividad en el interior de un edificio. Para ello contamos con una cámara estéreo Point Grey Bumblebee2, con la cual se extraerán varias secuencias de imágenes y a partir del procesamiento de éstas pretendemos obtener dichos resultados. De cada par de imágenes estéreo se extraerán una serie de puntos característicos, usando para ello los algoritmos de extracción de características SIFT[1] [3] y SURF[2][1]. Los puntos extraídos servirán como puntos de referencia para la estimación de la posición y del movimiento. Una vez hecho esto, se buscarán coincidencias entre los puntos obtenidos en dos pares de imágenes estéreo consecutivas temporalmente. Las coincidencias las utilizaremos para determinar, mediante la aplicación del modelo pin-hole en una cámara estéreo, las coordenadas reales (X,Y,Z) de los puntos obtenidos en las imágenes. Con el valor de la profundidad de un mismo punto en dos instantes consecutivos podremos obtener el desplazamiento del robot, y con dicho desplazamiento y la diferencia temporal entre la captura de ambas imágenes podremos estimar la velocidad que ha llevado el robot en su trayectoria.

En cuanto a la estructura de este documento, éste está compuesto por 6 capítulos y 2 apéndices. En el segundo capítulo se describe el estado del arte, en el cual se comenta, brevemente y de manera general, la situación en la que se encuentra hoy en día la navegación autónoma de robots por medio de técnicas de visión por computador, concretando para el caso en el que se

[1]Scale Invariant Feature Transform
[2]Speeded Up Robust Features

emplean cámaras de vídeo. En el tercer capítulo se hace una descripción del modelo Pin-Hole, que es el utilizado para determinar la correspondencia de un punto en el plano de la imagen obtenida con la cámara con su posición real en el espacio. En el cuarto capítulo se hace una introducción a la visión estereoscópica. Por su parte, en el quinto capítulo se introducen los principales métodos disponibles de extracción de características de imágenes, haciendo hincapié en las dos alternativas estudiadas en este proyecto: los descriptores SIFT y SURF. En este capítulo, además, se compararán ambas alternativas para justificar el procedimiento empleado finalmente. En el quinto capítulo se muestran los resultados obtenidos las diferentes situaciones que se ha probado el algoritmo implementado. En el último y sexto capítulo se incluyen las conclusiones y las expectativas futuras. Por último, en los apéndices se incluyen el procedimiento de cálculo del movimiento del robot y las características de la cámara empleada.

CAPÍTULO 2

ESTADO DEL ARTE

En este capítulo, hablaremos acerca de la situación actual y los avances realizados tanto en la detección de características como en la navegación autónoma de robots en los últimos años.

En primer lugar, debemos considerar que se entiende por navegación aquella metodología que permite guiar la trayectoria de un robot móvil a través de un entorno con la presencia de obstáculos. Con los años se ha pretendido que esta labor de navegación sea una tarea cada vez más propia del robot, de modo que la realice con total independencia del operador humano sin necesidad de la ayuda externa de éste. Esto es lo que se conoce como navegación autónoma. Este objetivo de que los robots puedan interactuar con su entorno y desplazarse por él a su antojo, durante muchísimos años ha parecido un propósito inalcanzable, pero poco a poco y gracias a los destacados avances realizados en áreas tan complejas como la inteligencia artificial, la lógica difusa o la visión por computador, han hecho que ese sueño imposible vaya poco a poco convirtiéndose en realidad. Algunos de los problemas principales que nos encontramos en la navegación autónoma son el correcto reconocimiento del entorno en el que el robot va a desempeñar su actividad, la localización de obstáculos (tanto si se mueven como si no) y la determinación de la posición

del robot tanto relativa como absoluta.

Para que el robot pueda superar dichos problemas y llevar a cabo esta labor de navegación con éxito, es necesario que tome consciencia del entorno que le rodea a través de una serie de sensores de modo que pueda conocer la posición de los obstáculos o el valor de una serie de parámetros internos que le ayuden en su trayectoria como son la distancia recorrida, la velocidad que lleva o su posición relativa en el espacio. Por tanto, a partir de la información proporcionada por los sensores será como se definirán los métodos y los algoritmos de navegación del robot.

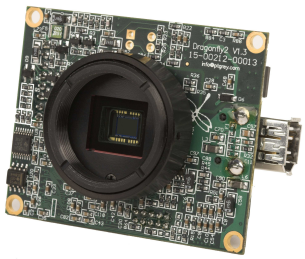
En cuanto a los sensores empleados por los robots para llevar a cabo una correcta navegación, hay que decir que éstos pueden ser muy diversos. Tenemos desde sensores de contacto o bumper, que se encargan de localizar por contacto directo la presencia de obstáculos en la trayectoria del robot, hasta sensores de infrarrojos, que se emplean en la detección de obstáculos o la medición del movimiento de las ruedas. En este proyecto nos centraremos en el problema de la determinación de la posición, concretamente en el cálculo de la distancia recorrida y la velocidad. Lo que se pretende con esto no es sólo que el robot sea capaz de alcanzar una posición final partiendo desde un punto inicial, sino que sea capaz de interactuar con su entorno pudiendo llevar a cabo tareas muy diversas, como por ejemplo el transporte de objetos de una posición a otra en el interior de un almacén.

Por lo tanto, para este proyecto, nos interesarán aquellos sensores empleados en *odometría*, que se define dentro de la navegación robótica como el estudio que nos permite determinar la posición del robot en el espacio, y, por consiguiente, nos permite estimar la velocidad llevada por éste y la distancia que ha recorrido. A continuación, se enumeran los diferentes sensores más comunes empleados en los sistemas odométricos:

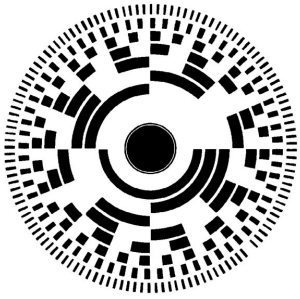
- Sensores de ultrasonidos.
- Cámaras digitales de video: monoculares y estéreo.
- Lasers.
- Ondas de radio.
- Balizas y GPS.
- Encoders.

- Acelerómetros.
- ...

(a) Láser



(b) Cámara CCD



(c) Encóder

(d) Baliza

Figura 2.1: Ejemplos de sensores empleados en odometría.

Tal y como dijimos en la introducción, el presente proyecto está estrechamente ligado a la visión por computador, por lo que nos interesará especialmente todo aquello que esté relacionado con el empleo de cámaras digitales en labores de localización y posicionamiento (SLAM[1]), o lo que también se conoce comúnmente como *odometría visual*, que se define como el proceso mediante el cual se determina la posición y la orientación de una cámara o un conjunto de cámaras por medio del análisis de una secuencia de imágenes

[1]Simultaneous Localisation and Mapping

adquiridas, sin tener ningún conocimiento previo del entorno. Al encontrarse la cámara o el sistema de cámaras acoplado al robot, el movimiento de éstas será solidario con el del robot y, por tanto, esta técnica nos servirá para determinar el movimiento del mismo.

Las principales ventajas que presentan los sistemas compuestos por cámaras digitales frente a los sistemas compuestos por otro tipo de sensores son las siguientes:

- Baratos.
- Sencillos.
- Salida digital.
- Permiten obtener gran cantidad de información del entorno.
- Permiten realizar una reconstrucción 3D del entorno.

Respecto al sistema de cámaras empleado, éste puede tratarse de un sistema monocular o un sistema estéreo. Los sistemas monoculares se caracterizan por estar conformados por una única cámara monocular (una lente) de modo que se toma en cada instante una única imagen del entorno, mientras que los sistemas estéreo están conformados por más de una cámara monocular (pueden estar o no alineadas) o una cámara de tipo estéreo (más de una lente), de forma que en cada instante se adquieren varias imágenes de la escena desde ángulos o puntos de vista diferentes, siendo conocida la posición relativa entre las cámaras o, en el caso de la cámara estéreo, la distancia entre las lentes. Las dos principales diferencias entre ambos sistemas son las siguientes:

1. En el sistema monocular es necesario que el robot se mueva para poder estimar la profundidad, mientras que en el sistema estéreo no es necesario que haya movimiento.
2. El sistema monocular es más complejo, ya que requiere un mayor número de transformaciones para estimar la posición, mientras que el sistema estéreo necesita únicamente un par de transformaciones geométricas sencillas.

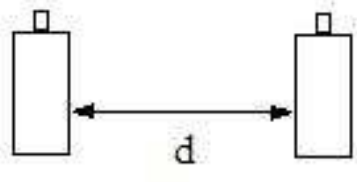


Figura 2.2: Sistema de cámaras estéreo. Donde d es la posición relativa entre las cámaras.

Como hemos dicho anteriormente, mientras que en un sistema de cámaras estéreo nos basta con realizar una serie de transformaciones geométricas y conocer el valor de la disparidad para estimar el valor de la profundidad de los objetos que se encuentran en la imagen [7], en un sistema monocular será necesario realizar el cálculo del flujo óptico de la misma, el cual se entiende como el movimiento aparente resultante en la imagen si la cámara, o uno o varios objetos de la escena 3D, se mueven. El sistema a emplear en nuestro proyecto será un sistema estéreo, por lo que no vamos a profundizar en el cálculo del flujo óptico, pero si se desea obtener más información acerca de la obtención de la profundidad a partir de él, puede consultarse el artículo publicado por Shahraray et al [28].

Una vez hemos considerado las diferentes configuraciones del sistema de cámaras que podemos emplear, debemos tener en cuenta que en dichos sistemas de visión artificial es clave la detección de puntos estacionarios en las imágenes, es decir, puntos que permanezcan invariantes dentro de la secuencia de imágenes. Por tanto, la etapa de detección de puntos de interés o características es crítica para poder realizar con éxito la estimación del movimiento del robot. A continuación, se deberá realizar un correcto emparejamiento ("matching") de los puntos de interés detectados en varias imágenes consecutivas, de modo que se produzcan el menor número de enlaces falsos posibles. Una vez hecho esto, procederemos a posicionar dichos puntos de interés en el espacio tridimensional, para a partir de ahí estimar la distancia recorrida y la velocidad.

Como se ha dicho en el párrafo anterior, la detección de características es una etapa clave dentro del sistema que queremos implementar. Realizar una buena detección en la que los puntos encontrados sean lo suficientemente distintivos y que den pie al menor número de errores posibles en la etapa de emparejamiento, es fundamental para llevar a cabo tareas de localización y posicionamiento de manera exitosa y precisa. Por tanto, lo primero que

deberíamos plantearnos es, ¿qué es un punto característico de una imagen? Se trata de aquellos puntos que difieren considerablemente de sus vecinos, estando esa diferencia asociada a uno o varios cambios en alguna propiedad de la imagen. Las propiedades que normalmente se analizan son la intensidad, el color y la textura.

Desde que comenzase a aumentar el interés por el uso de cámaras digitales como sensor principal en las aplicaciones de SLAM (debido a ello también es conocido como Visual SLAM), el número de detectores y avances desarrollados con relación a la extracción de características de imágenes ha aumentado considerablemente. El detector más usado hasta el momento es probablemente el detector de esquinas de Harris [17], que fue propuesto en 1988 y se basaba en los autovalores de una matriz compuesta por los gradientes de la imagen en las direcciones x e y. El principal problema de este detector es que las esquinas detectadas no son invariantes a las variaciones de escala. Por ello, en 1998, Lindeberg presenta el concepto de selección automática de escala[21], que permite detectar puntos de interés en una imagen con su propia escala característica.

En 1997, Smith y Brady presentan el detector de esquinas SUSAN[2] [19], el cual, en vez de evaluar los gradientes locales como hacía Harris, emplea un enfoque morfológico en el que se compara cada píxel con un área concreta de sus vecinos y, en función de la semejanza en intensidad del área, se determina si es una esquina o un borde. Posteriormente, Mikolajczyk y Schmid trabajaron sobre el método de Lindeberg, creando un detector robusto, invariante ante cambios de escala y con gran repetibilidad de los puntos de interés hallados, al que llamaron $Harris - Laplace$ y $Hessian - Laplace$ [22]. Ellos empleaban una muestra de Harris o el determinante de la matriz Hessiana (de ahí los nombres Harris-Laplace y Hessian- Laplace) para seleccionar la posición y el operador laplaciano para elegir la escala.

Por su parte, Lowe, en 1999, simplificó el Laplaciano de una Gaussiana (LoG) con un filtro Diferencia de Gaussianas (DoG), con lo que obtuvo un detector con mejor rendimiento y mayor velocidad de computación, conocido como SIFT [4]. El detector descrito por Lowe ha sido uno de los más empleados hasta el momento y es uno de los que proporciona mejor rendimiento. Lowe ha continuado trabajando sobre su propio detector [3], consiguiendo obtener descriptores aún más característicos y menos sensibles a posibles distorsiones de la imagen. Además, sobre el detector SIFT se han realiza-

[2]Smallest Univalue Segment Assinilating Nucleus

do numerosísimos trabajos como el llevado a cabo en 2004 por Ledwich y Williams [25], en el que se describe un método para reducir la complejidad y el tiempo de emparejamiento de los descriptores SIFT en aplicaciones de localización en el interior de edificios.

Por último, en 2008, Herbert Bay, Andreas Ess, Tinne Tuytelaars y Luc Van Gool introducen el detector SURF [1], que es, a día de hoy, una referencia y al que diversos estudios [20, 18, 14] definen como el detector que presenta mejor rendimiento en la actualidad. Este detector, basado en una aproximación de la matriz Hessiana, guarda algunas similitudes con el SIFT, pero introduce mejoras considerables en el rendimiento, disminuyendo la velocidad de computación (gracias al empleo de imágenes integrales) y aumentando la robustez del detector sin afectar negativamente a la singularidad de los descriptores, que continúan siendo suficientemente distintivos.

Todos estos detectores que hemos nombrado en el párrafo anterior han sido empleados en numerosas aplicaciones que van desde la identificación de objetos o el reconocimiento facial hasta labores de localización y creación de mapas de reconstrucción del entorno (SLAM). Algunos ejemplos destacados de estas aplicaciones son los siguientes:

- En [4], Lowe emplea los descriptores SIFT para llevar a cabo el reconocimiento de objetos almacenados previamente en una base de datos.
- Bay, Fasel y Van Gool [24], desarrollan una aplicación para, mediante el empleo de los descriptores SURF, realizar el reconocimiento de objetos de arte en el interior de un museo.
- Murillo et al [26], emplean los descriptores SURF en un sistema de localización basado en visión. En su artículo hacen referencia a la buena precisión y al gran rendimiento obtenido con dichos descriptores.
- Srivastava y Goyal [27], desarrollan para un proyecto de la Universidad de Stanford una técnica de reconocimiento de cuadros basada en la implementación de los descriptores SIFT de Lowe.
- En [30], Se, Lowe y Little presentan una aplicación de SLAM visual basada en descriptores SIFT, el sistema es implantado en un robot equipado con una cámara estéreo trinocular.

Por otro lado, dentro de las aplicaciones de SLAM, existen también diversas técnicas basadas en otros métodos diferentes a los detectores de interés

descritos arriba. Si se desea profundizar en algunos de estos trabajos, se pueden consultar los realizados por Thrun et al [31], en el que se describe un método de localización probabilístico, o Jens-Steffen Gutmann y Dieter Fox [32, 33], en los cuales se realiza una comparación de algunos de estos métodos de localización mencionados anteriormente. Aún así, estos métodos no son de ningún modo incompatibles con el empleo de los descriptores SIFT o SURF, ya que algunos investigadores han desarrollado aplicaciones de SLAM que combinan ambas técnicas como, por ejemplo, el artículo presentado en 2005 por Arturo Gil et al [34].

Finalmente, en cuanto a la dirección que deben tomar las investigaciones futuras en esta área, éstas deberían estar enfocadas a lograr una mayor estabilidad, un mayor grado de invarianza de dichos puntos y una menor sensibilidad al ruido en cuanto a los puntos de interés detectados. Esto debería permitir desarrollar algoritmos más rápidos, más precisos, más sencillos y, por tanto, con un mejor rendimiento que los actuales.

CAPÍTULO 3

SISTEMA DE VISIÓN EMPLEADO

3.1. Modelo PIN-HOLE

En esta sección se va a describir el modelo utilizado para determinar la correspondencia de cada punto del plano de la imagen (2D) con su posición espacial en la realidad (3D). En este caso, el modelo utilizado es el Pin-Hole. Se trata de un modelo muy sencillo, siendo el más usado en la práctica a la hora de analizar cámaras CCD (la relativa a este proyecto).

Este modelo supone que para todo punto de la imagen, el rayo de luz que sale rebotado del objeto y llega al sensor de la cámara va a atravesar un único punto, independientemente de cuál sea el punto de origen y cuál sea el punto de impacto en la imagen.

Dado que el modelo pin-hole es un modelo ideal en el cual no se tienen en cuenta las distorsiones de la lente, podríamos considerar la utilización de parámetros que ajusten mejor su comportamiento a la realidad. Para el presente proyecto vamos a considerar que el error cometido es muy pequeño, por lo que no se va a introducir ningún factor de corrección.

En este modelo el sistema de referencia de la cámara está situado en el centro de la proyección de la imagen, coincidiendo, por tanto, el eje Z del sistema con el eje óptico. El plano de proyección de la imagen se encuentra situado a una distancia equivalente a la distancia focal, de forma perpendicular al eje óptico. El punto de intersección entre el plano imagen y el eje óptico se denomina punto principal (p).

En la siguiente figura podemos ver gráficamente el esquema del modelo:

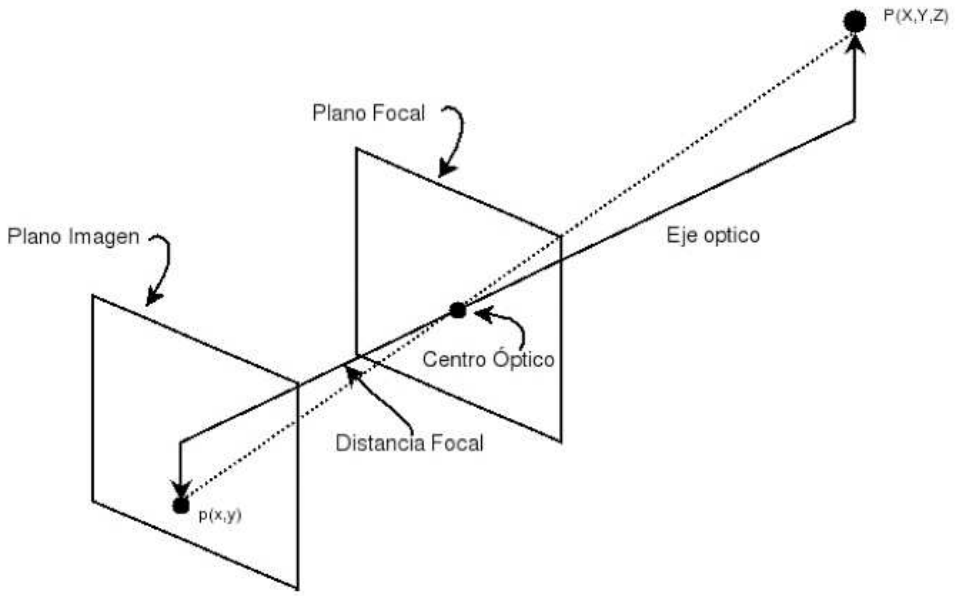


Figura 3.1: Modelo Pin-Hole

Siendo la representación matemática del modelo la siguiente:

$$x = f\frac{X}{Z} \tag{3.1}$$

$$y = f\frac{Y}{Z} \tag{3.2}$$

Donde f es la distancia focal[1] de la cámara, (x,y) son las coordenadas del

[1]La distancia focal es la distancia que separa el centro óptico del plano de proyección

punto de interés en la imagen y (X,Y,Z) son las coordenadas del punto de interés en el espacio.

Esto mismo puede verse de forma matricial de la siguiente manera:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.3}$$

Para el modelo de nuestro trabajo debemos considerar que, en realidad, el eje de coordenadas del plano de la imagen no es el centro óptico de la misma (ver figura 3.2), sino que éste se encuentra desplazado respecto a dicho punto. Concretamente, se encuentra en la esquina superior izquierda de la imagen. Por tanto, en realidad, la representación matricial queda de la siguiente manera:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & px \\ 0 & f & py \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.4}$$

Donde px y py son las coordenadas del punto principal en el sistema de referencia de la imagen. Matemáticamente esto queda escrito de la siguiente manera:

$$x = f\frac{X}{Z} + px \tag{3.5}$$

$$y = f\frac{Y}{Z} + py \tag{3.6}$$

de la imagen.

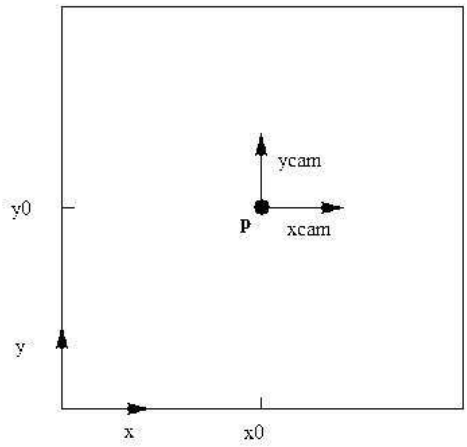


Figura 3.2: Sistema de coordenadas de la imagen (x,y) y sistema de coordenadas de la cámara (xcam,ycam).

Por otro lado, podríamos encontrarnos que las coordenadas (X,Y,Z) del punto de interés estén dadas respecto a un eje de coordenadas externo, en cuyo caso nos encontraríamos con que, para poder aplicar el modelo pin-hole, tendríamos que realizar la transformación de dichas coordenadas al sistema de referencia de la cámara. Para realizar dicha transformación, tendremos que introducir en el modelo la correspondiente matriz de rotación y traslación. Quedando el modelo entonces de la siguiente manera:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & px \\ 0 & f & py \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix} \begin{bmatrix} tx \\ ty \\ tz \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.7}$$

En nuestro caso, nos vamos a encontrar que el sistema de coordenadas de la cámara se encuentra alineado con el sistema de coordenadas global.

Por último, al aplicar este modelo en las cámaras CCD (descritas en la sección 3.2) tendremos que realizar una última consideración, ya que debemos tener en cuenta que las distancias están expresadas en píxeles, por lo que habrá que utilizar un factor de conversión λ que nos permita pasar de píxeles a metros (ver expresión 3.16).

3.2. Cámaras CCD

En primer lugar, debemos tener en cuenta que la cámara o sensor de imagen es el elemento que se encarga de recoger la información de la escena de interés, transmitiéndola posteriormente al ordenador. Las cámaras CCD[2] son aquellas que utilizan como elemento sensor un dispositivo de carga acoplada o CCD, el cual está conformado por diminutas células fotoeléctricas en las que se registra la imagen.

La capacidad de resolución o detalle de la imagen depende del número de células fotoeléctricas del CCD, las cuales son las encargadas de captar los rayos luminosos. El número de células fotoeléctricas se expresa en píxeles, siendo éste el elemento básico de los CCD. A mayor número de píxeles, mayor resolución tendrá la cámara.

Dependiendo del color de la imagen de salida, este tipo de cámaras pueden ser:

- Cámaras de blanco y negro.
- Cámaras de color.

La principal diferencia entre ambos tipos de cámaras está en que para las imágenes en color es necesario que por cada píxel de la imagen existan tres en la CCD, para, de este modo, poder recoger la información de las componentes roja, verde y azul (RGB[3]) de la imagen, con lo que seremos capaces de captar cualquier color de la misma. Para mejorar la resolución y la calidad de la imagen, existen otros tipos de cámaras a color que emplean tres sensores CCD acoplados (uno para cada color), pero el coste de estos dispositivos es mucho mayor.

En cuanto a los parámetros más importantes que debemos tener en cuenta a la hora de utilizar este tipo de cámaras, éstos son los siguientes:

Shutter

[2]Charge Coupled Device
[3]Red-Green-Blue

Es el periodo de tiempo en el que los elementos de la CCD están bajo los efectos luminosos.

Tiempo de adquisición

Tiempo necesario para que el CCD transmita la información captada.

Factor Gamma

Normalmente se busca que la relación entre la señal de entrada y la de salida sea lineal. En las cámaras CCD buscaremos que la relación entre el número de fotones y los voltios de salida sea también lineal. Sin embargo, esto no se cumple y por medio del factor gamma se indica la no linealidad del sistema.

Sensibilidad absoluta

Es la iluminación mínima necesaria para que la cámara produzca una salida.

Sensibilidad relativa

Es el número de fotones necesarios para que la cámara pase de un valor al siguiente.

Relación señal ruido

Este valor nos sirve de indicador del nivel de interferencias de la señal de salida.

Relación de los píxeles

Los píxeles pueden ser cuadrados o rectangulares, con relaciones x/y igual a 1,1 o $\frac{4}{3}$ para adecuarse a los diferentes estándares de televisión.

Ganancia

Se trata de la ganancia de la cámara frente a la longitud de onda de la luz que incide sobre la CCD. La totalidad de cámaras digitales tienen una mayor ganancia en la zona del infrarrojo, de modo que la imagen de salida tenga el mayor parecido posible a la vista por el operador humano.

Finalmente, si se quiere obtener más información detallada acerca de este tipo de cámaras, puede consultarse [8].

3.3. Visión Estereoscópica

Como ya hemos dicho anteriormente, la cámara empleada en el proyecto es una cámara de tipo estéreo, por lo que en este apartado vamos a considerar las diferentes características propias de este tipo de sistemas, que emulan el sistema de visión humano.

En primer lugar, debemos tener presente que los sistemas de visión estéreo permiten, a partir de la combinación de una secuencia de imágenes de una escena (bastará con un par de imágenes), la extracción de información tridimensional y, de manera muy sencilla, el cálculo de la profundidad de cada uno de los puntos encontrados en las imágenes. Para ello, simplemente deberemos tener en cuenta una serie de consideraciones geométricas que nos permitirán reconstruir el entorno tridimensional (3D) a partir de imágenes bidimensionales (2D). Las diferencias existentes entre las distintas imágenes se deberán principalmente al movimiento relativo del sensor o al movimiento realizado por los objetos de la escena.

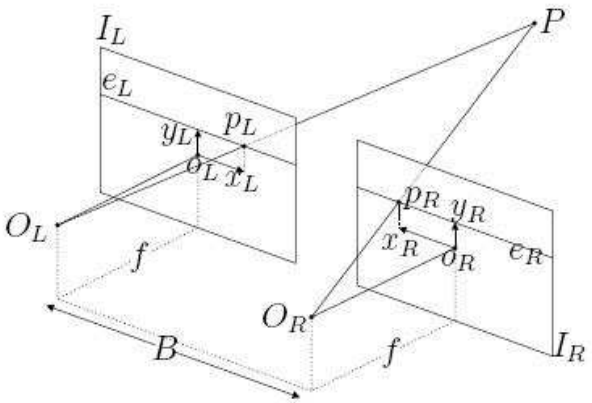


Figura 3.3: Configuración del par estéreo. Imagen extraída de [12].

Para poder extraer la información tridimensional de la escena y realizar el cálculo de la profundidad, debemos realizar previamente los siguientes pasos (se explican con más detalle en el capítulo 4):

- Extracción de puntos característicos en las diferentes imágenes.
- Emparejamiento de los distintos puntos hallados en cada una de las imágenes.

Una vez hemos realizado los dos pasos anteriores, podemos proceder al cálculo de la profundidad. Para ello debemos tener en cuenta la relación geométrica del par estéreo (ver figua 3.4).

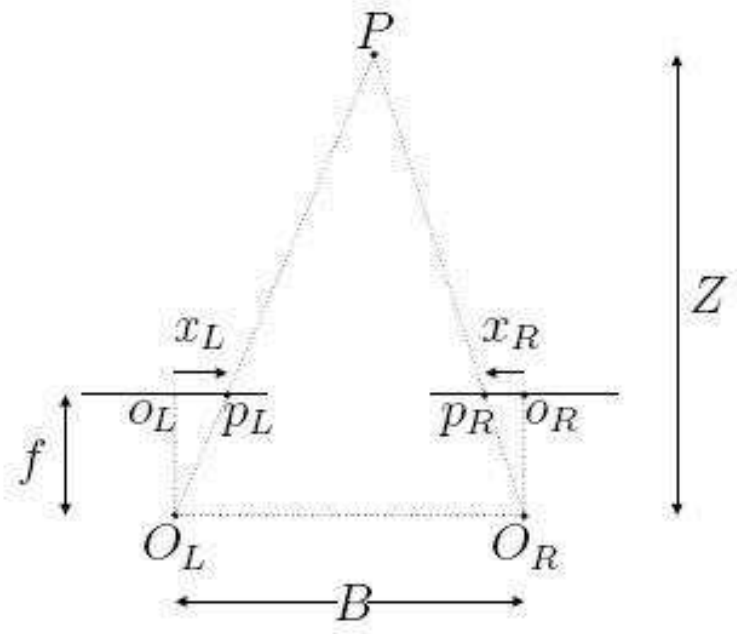


Figura 3.4: Relación geométrica del par estéreo. Imagen extraída de [12].

Donde, B es la línea de base o la distancia entre los dos centros ópticos de la cámara (distancia entre las dos lentes), f es la distancia focal, O_L y O_R son los centros ópticos de la lente izquierda y la lente derecha respectivamente, p_L y p_R son las proyecciones del punto P en el plano de la imagen izquierda y derecha, y x_L y x_R son las posiciones en el eje x de las proyecciones del punto P en ambas imágenes.

Llegados a este punto, es especialmente importante introducir el concepto de *disparidad d*. La disparidad se define como la diferencia entre las coordenadas x (horizontales) de las proyecciones del punto P en ambas imágenes. Esto

queda de la siguiente manera (dependiendo del sistema de referencia utilizado, podría verse alterada la definición, de modo que el valor de la disparidad quede siempre positivo):

$$d = x_L - x_R \tag{3.8}$$

Además, debemos tener en cuenta la relación entre las coordenadas de las proyecciones del punto P en ambas imágenes:

$$x_L = d + x_R \tag{3.9}$$

$$y_L = y_R \tag{3.10}$$

Teniendo en cuenta esta relación y que la cámara cumple con el modelo Pin-Hole (descrito en la sección 3.1), podemos determinar la profundidad Z del punto P(X,Y,Z) de la escena, haciendo uso únicamente de las semejanzas entre triángulos (teniendo como referencia la imagen 3.4).

Del modelo Pin-Hole, podemos extraer las coordenadas X e Y :

$$x_L = f\frac{\frac{B}{2}}{Z} \tag{3.11}$$

$$x_R = f\frac{-\frac{B}{2}}{Z} \tag{3.12}$$

$$y_R = y_L = f\frac{Y}{Z} \tag{3.13}$$

Con esto ya podemos obtener la relación entre las coordenadas del punto P y la disparidad:

$$\begin{aligned} d &= x_L - x_R \\ &= f\frac{\frac{B}{2}}{Z} - f\frac{-\frac{B}{2}}{Z} \end{aligned}$$

$$\begin{aligned} &= \frac{f}{Z}(\frac{B}{2}+\frac{B}{2}) \\ &= f\frac{B}{Z} \end{aligned} \tag{3.14}$$

O lo que es lo mismo, podemos obtener el valor de la profundidad del punto P:

$$Z = f\frac{B}{d(x_L, x_R)} \tag{3.15}$$

Tal y como podemos ver, la profundidad es un valor inversamente proporcional a la disparidad, es decir, a mayor valor de profundidad tenemos un valor menor de disparidad y viceversa. Sin embargo, debemos tener en cuenta que, tal y como ya se dijo anteriormente, el valor de la disparidad d viene dado en píxeles y el valor de la profundidad Z queremos calcularlo en metros. Por tanto, debemos hacer uso de un factor de conversión λ que nos permita pasar las unidades de píxeles a metros.

$$\lambda = \frac{ancho\ del\ CCD\ en\ metros}{ancho\ de\ la\ imagen\ en\ pixeles} \tag{3.16}$$

Incluyendo este factor de conversión λ en la expresión de la profundidad Z, ésta, finalmente, nos queda de la siguiente manera:

$$Z = f\frac{B}{d(x_L, x_R)\lambda} \tag{3.17}$$

Por último, diremos cómo calcular el ancho del sensor CCD en metros. Para calcularlo debemos consultar, en primer lugar, las especificaciones de nuestra cámara para ver sus dimensiones y su relación $\frac{x}{y}$. En nuestro caso, las dimensiones (ver figura 3.5) son de $\frac{1}{3}''$ y la relación $\frac{x}{y} = \frac{4}{3}$. Con dicha relación, si despejamos la expresión 3.18 obtenemos el valor de x e y.

$$x^2 + y^2 = (\frac{1}{3}'')^2 \tag{3.18}$$

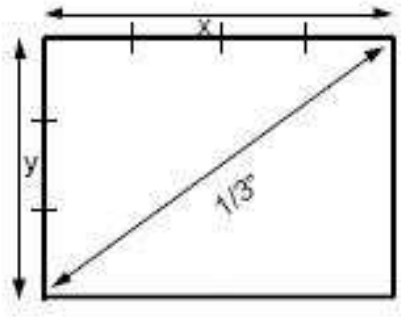


Figura 3.5: Dimensiones del CCD de la cámara estéreo empleada.

CAPÍTULO 4

DETECTORES DE PUNTOS DE INTERÉS

En este capítulo se van a describir los principales métodos utilizados para la extracción de características en aplicaciones de SLAM. Aquellos puntos que vayan a ser localizados deben ser lo más estables posibles y permanecer invariantes ante posibles cambios de orientación, rotación o escala de las imágenes, de modo que dichos puntos siempre sean encontrados entre un buen número de imágenes consecutivas. Estas condiciones son absolutamente necesarias para poder desarrollar un buen algoritmo de localización. Además, los puntos extraídos deben ser lo suficientemente característicos para que puedan ser correctamente asociados aquellos encontrados en dos imágenes diferentes, aunque éstas hayan sido tomadas desde posiciones diferentes. Es importante tener en cuenta que la mayoría de los detectores están pensados para tratar imágenes con textura, ya que no aparecen descriptores en imágenes uniformes. Por tanto, las principales características que deben tener los puntos de interés son las siguientes:

- Deben ser fáciles de extraer.

- Deben ser lo suficientemente característicos como para que permitan una correcta identificación y produzcan el menor número de errores posible.
- Invariantes al ruido de la imagen, cambios de iluminación, escala, rotación y orientación.
- Fáciles de reconocer y de enlazar con los puntos extraídos en otra imagen.

A continuación, se definirán de manera general y se evaluarán los métodos más importantes que han sido empleados por los investigadores hasta el día de hoy. A la hora de definirlos se profundizará en mayor medida en aquellos procedimientos que han sido considerados de mayor relevancia para este proyecto: los algoritmos SIFT y SURF.

4.1. Detector de esquinas de Harris

Este método [17] ha sido el más empleado hasta el momento. Fue definido por los investigadores Chris Harris y Mike Stephens en 1988.Los puntos detectados por este procedimiento permanecen invariantes ante cambios en la escala, rotación, iluminación y ruido de la imagen. Este detector define la siguiente matriz C(x,y), que se calcula sobre una subventana de tamaño nxn para cada punto de interés en la posición (x,y).

$$C(x,y) = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \tag{4.1}$$

Donde I_x e I_y son los gradientes vertical y horizontal de la imagen. Si definimos a λ_1 y λ_2 como los autovalores de la matriz descrita arriba, podemos definir entonces la función de autocorrelación R como:

$$R = \lambda_1 \lambda_2 - K(\lambda_1 + \lambda_2) \tag{4.2}$$

Donde K es un parámetro experimental. En esta función R vamos a distinguir tres casos posibles:

- La función tendrá un máximo si ambos autovalores son altos, lo que quiere decir que un desplazamiento en cualquier dirección va a producir un incremento importante, indicando, por tanto, que se trata de una esquina.

- La función será casi nula si ambos autovalores son bajos, es decir que un desplazamiento en cualquier dirección va a producir cambios muy pequeños, por tanto, la región que engloba la subventana de tamaño nxn es de intensidad constante (pertenece al mismo objeto).

- Si un autovalor es alto y el otro bajo, entonces, la función tendrá forma de cresta. Por tanto, sólo los desplazamientos en una dirección van a producir pequeños cambios en C(x,y) y cambios significativos en la dirección perpendicular. Esto indicará la presencia de un borde.

4.2. Detector de Harris-Laplace

Los puntos de interés localizados con este método son invariantes ante cambios de rotación y escala. Dichos puntos son detectados por medio de una función de Harris seleccionada en el espacio de escalas por el operador Laplaciana. La escala seleccionada es la que va a determinar el tamaño de la región de interés estudiada. Este detector se caracteriza porque detecta características localizadas espacialmente de manera más precisa, cosa que es deseable cuando estos puntos son utilizados en labores de localización y reconstrucción y no sólo en funciones de reconocimiento.

4.3. Detector de SUSAN

El detector de esquinas de SUSAN fue desarrollado por S. M. Smith y J. M. Brady [19]. Se trata de una técnica diferente a la empleada en el detector de Harris, ya que en vez de realizar la evaluación de gradientes locales, este detector lo que hace es emplear un enfoque morfológico. Se trata de una técnica de procesamiento de imágenes de bajo nivel, que, aparte de para la detección de esquinas, ha sido utilizada en la detección de bordes y la eliminación de ruido.

Este detector funciona de la siguiente manera: Para cada píxel de la imagen consideramos los vecinos en el interior de un círculo de radio fijo alrededor de él. El píxel central es referido como el núcleo y su valor de intensidad es tomado como referencia. Entonces, todos los píxeles en el interior de ese círculo son clasificados en dos grupos: los de intensidad parecida al núcleo o los que poseen un valor de intensidad diferente. De este modo, cada punto de la imagen tiene asociada un área local de brillo similar, cuyo tamaño está asociado a información importante sobre la estructura de la imagen en dicho punto. En aquellas partes de la imagen más o menos homogéneas, el área local de brillo similar cubre casi la totalidad del círculo de vecindad. Al encontrarnos en presencia de un borde, el área cubre alrededor del 50 % del círculo y en presencia de una esquina este porcentaje cae por debajo del 25 % (ver figura 4.1. Por tanto, las esquinas pueden ser localizadas como los puntos de la imagen en los que el número de píxeles de intensidad similar en el círculo de vecindad es mínimo y está por debajo de un cierto umbral predefinido. Para darle mayor robustez al método, los píxeles más próximos al núcleo reciben un mayor valor de ponderación. Además, se declaran una serie de reglas para eliminar aquellos descriptores que no interesan. Finalmente, los mínimos locales son los seleccionados como candidatos.

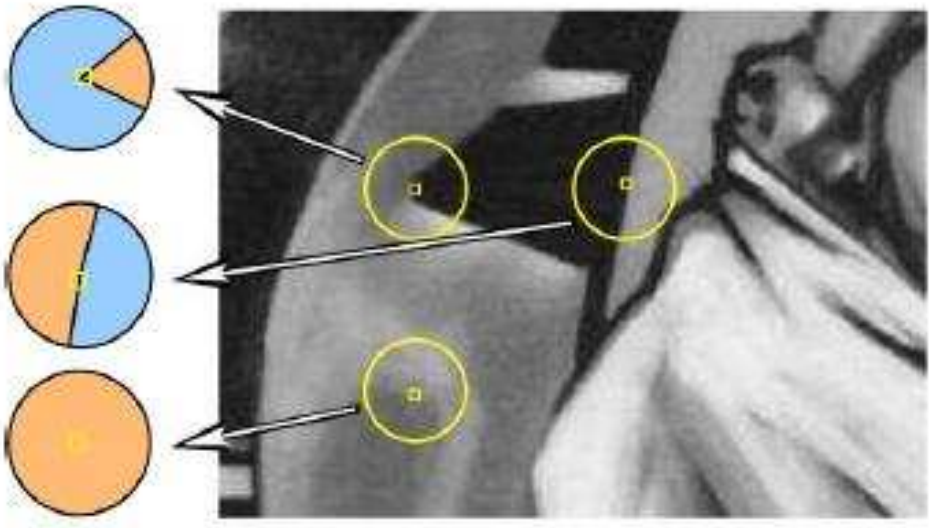


Figura 4.1: Detector de Susan. Distinción entre un área homogénea, un borde o una esquina. Imagen extraída de [20].

4.4. Detector SIFT

Este detector [4] fue presentado por el investigador y profesor de la Universidad British Columbia, David G. Lowe, en el año 1999. Los puntos de interés encontrados con este método se caracterizan por permanecer invariantes ante los posibles cambios producidos en los siguientes parámetros de nuestro sistema:

- Iluminación.
- Ruido de la imagen.
- Rotación.
- Escala.
- Pequeños cambios de orientación.

En cuanto a las características de este detector, éste se caracteriza principalmente por generar una gran cantidad de descriptores muy estables y por presentar un buen rendimiento en relación a la velocidad de cálculo y precisión.

Respecto a las etapas en las que se divide este algoritmo, éstas son las siguientes:

1. Detección de máximos y mínimos en el espacio escala.
2. Localización y filtrado de los puntos de interés.
3. Asignación de orientación.
4. Generación de los descriptores de los puntos clave.

A continuación, se describirá con detalle cada una de estas etapas.

4.4.1. Detección de máximos y mínimos en el espacio escala

Para la detección de los puntos de interés vamos a aplicar una serie de filtros en cascada.

El primer paso a la hora de detectar los puntos de interés es identificar las diferentes posiciones y escalas que pueden asignarse repetidamente bajo diferentes puntos de vista del mismo objeto. Cada imagen va a ser filtrada con una Gaussiana, de modo que los puntos de interés de los descriptores SIFT corresponden a los máximos y mínimos locales resultantes de obtener la diferencia entre varios filtros Gaussianos a escalas diferentes. De modo que el espacio escala de una imagen se define como la función $L(x, y, \sigma)$, resultante de la convolución de la imagen de entrada con una Gaussiana de escala variable:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{4.3}$$

Donde tenemos que I es la imagen de entrada y G es la Gaussiana de escala variable, la cual es igual a:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{\sigma^2}} \tag{4.4}$$

Para lograr una detección eficiente de puntos de interés estables, este detector sugiere la utilización en el espacio-escala de la convolución de una diferencia de Gaussianas con la imagen. De modo que esto queda de la siguiente manera:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{4.5}$$

Donde $D(x, y, \sigma)$ es la diferencia de los resultados de difuminar la imagen de entrada con una Gaussiana de escala kσ y σ.

Los motivos que llevan a la utilización de esta función D (también llamada DoG[1]) son por un lado su mayor eficiencia al tratarse simplemente de la resta de dos imágenes, y por otro lado debido a que nos da una aproximación más precisa del Laplaciano normalizado de una Gaussiana.

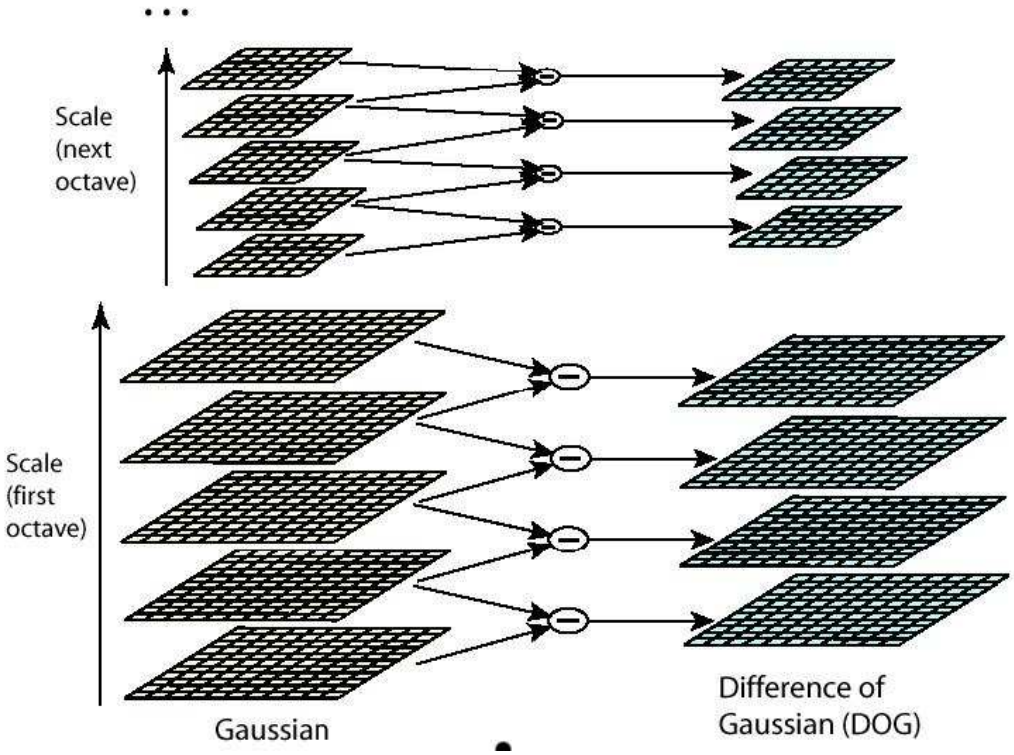


Figura 4.2: Diagrama que muestra las imágenes difuminadas a diferentes escalas, y las imágenes resultantes de aplicar la diferencia de Gaussianas. Imagen extraída de [3].

Por tanto, el primer paso en la detección de los puntos de interés es la convolución de la imagen con filtros Gaussianos de diferentes escalas, y la generación de las imágenes DoG(diferencia de Gaussiana) obtenidas de aplicar la diferencia entre dos imágenes borrosas adyacentes.

Las imágenes resultantes de la convolución son agrupadas por octava[2](cada octava es dividida en un número s de intervalos), y el valor del factor k (determina la separación de las imágenes en el espacio-escala y es definido como $k = 2^{\frac{1}{s}}$) es seleccionado de manera que obtengamos un número fijo de

[1] $Difference\ of\ Gaussians \equiv Diferencia\ de\ Gaussianas$

[2] Una octava corresponde a doblar el valor de σ

imágenes borrosas por octava. Además, esto nos asegura que el número de imágenes resultantes de la diferencia de Gaussianas sea el mismo para cada octava (s+3). Después de procesar cada octava, la imagen Gaussiana, a la que le corresponde un valor de σ igual al doble del valor inicial, es reducida por un factor igual a 2 y el proceso es repetido.

Los puntos de interés o puntos clave son identificados como los máximos o mínimos locales de las imágenes DoG a través de las diferentes escalas. Cada píxel en las imágenes DoG es comparado con sus 8 vecinos en su misma escala, más los correspondientes 9 vecinos en las escalas vecinas (superior e inferior). Si el píxel es un máximo o un mínimo local, entonces, es seleccionado como un posible punto de interés o punto clave.

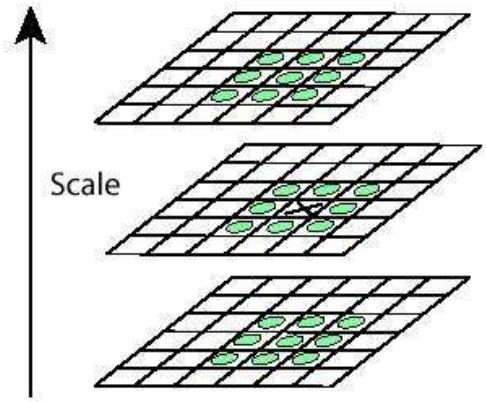


Figura 4.3: Proceso de comparación entre diferentes escalas. Imagen extraída de [3].

Con esto ya tendríamos un primer grupo de posibles candidatos como puntos de interés.

4.4.2. Localización de los puntos clave

En esta etapa eliminamos y descartamos aquellos puntos de interés que no sean adecuados, ya sea por su bajo contraste o su mala localización (por ejemplo en un borde).

Para cada candidato a punto de interés debemos hacer lo siguiente:

- Mediante la interpolación de los datos cercanos se determina de manera precisa su posición (para más información de este punto ver [3]).

- Aquellos con bajo contraste deben ser eliminados.

- Aquellos que correspondan a bordes deben también eliminarse.

- A cada punto de interés hay que asignarle una orientación (siguiente etapa).

4.4.3. Asignación de orientación

En esta etapa, otorgándole a cada punto de interés una orientación se logra dotar a los puntos de invarianza a la rotación de la imagen.

Para determinar la orientación del punto clave, se calcula un histograma del gradiente de la orientación en los puntos vecinos al punto de interés (utilizando para ello la imagen Gaussiana de escala más cercana a la del punto de interés). Para cada imagen muestreada la magnitud del gradiente (m(x,y)) y la orientación ($\theta(x, y)$) es la siguiente:

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2} \quad (4.6)$$

$$\theta(x, y) = tan^{-1} \frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)} \quad (4.7)$$

A la contribución de cada píxel vecino se le asigna un peso en función de la magnitud del gradiente y a una Gaussiana con σ igual a 1.5 veces la escala del punto clave o punto de interés. Los picos del histograma corresponderán a las orientaciones dominantes. Se crea aparte un punto de interés con la dirección relativa al máximo del histograma y con cualquier dirección que tenga al menos el 80 % del valor máximo.

4.4.4. Generación de los descriptores de los puntos clave

En las etapas anteriores hemos asignado a cada punto de interés una posición, una escala y una orientación. Por tanto, en esta etapa lo que haremos será crear los descriptores usando para ello histogramas de orientaciones. Los descriptores creados deberán ser suficientemente distintivos y permanecer dentro de lo posible invariantes ante los cambios producidos en la imagen.

Para darle a los descriptores invarianza ante la orientación, todas las propiedades de los puntos de interés son medidas en relación a su orientación

El descriptor se computa como un conjunto de histogramas de orientación creados sobre una región de muestra 4x4 en los píxeles vecinos. Los histogramas de orientación se construyen en función de la orientación del punto clave, proviniendo los datos de orientación de la imagen Gaussiana más cercana en escala a la del punto de interés. Al igual que antes, la contribución de cada píxel es ponderada por la magnitud del gradiente y por una Gaussiana de escala 1.5 veces la del punto de interés.

Cada histograma contiene 8 referencias, y cada descriptor contiene un conjunto de 4 histogramas alrededor del punto característico. Esto nos lleva a que el descriptor SIFT está formado por un vector de tamaño 4x4x8 = 128 elementos. La normalización de este vector mejora la invarianza ante cambios en la iluminación del entorno.

Finalmente, cada descriptor contendrá la siguiente información:

- Posición (x,y) en la imagen.
- Orientación.
- Escala.
- Descripción de su entorno por medio de un conjunto de gradientes.

4.4.5. *Matching*

Finalmente, el último paso que realizará nuestro detector será el emparejamiento de los puntos de interés hallados en dos imágenes consecutivas. Dicho emparejamiento lo hará identificando al vecino más cercano, que se define como el punto de interés con la mínima distancia Euclídea para el descriptor. En este punto, se aplicarán diversos métodos para evitar que se produzcan falsos emparejamientos.

4.5. Detector SURF

Este detector fue presentado en 2008 por Herbert Bay et al [1]. Aunque guarda cierta similitud con el detector SIFT, el detector SURF presenta notables diferencias con éste. En comparación con SIFT, sus autores afirman que este detector presenta principalmente dos mejoras:

- Velocidad de cálculo considerablemente superior.
- Mayor robustez ante posibles transformaciones de la imagen.

Para conseguir estas mejoras, el detector SURF reduce la dimensión y la complejidad de los descriptores obtenidos, pero lo hace de modo que éstos continúen siendo suficientemente característicos e igualmente repetitivos.

A continuación, vamos a describir los pasos necesarios para obtener los descriptores SURF.

4.5.1. Detección de puntos de interés

Este detector está basado principalmente en la matriz Hessiana, ya que usa una aproximación muy básica de ésta. El motivo de usar la matriz Hessiana se encuentra en su buen rendimiento en cuanto a la velocidad de cálculo y la precisión. Sin embargo, este detector, al contrario del método empleado por otros detectores, en vez de usar una medida diferente tanto para elegir la posición como la escala, lo que hace es emplear el determinante

de la matriz Hessiana en ambos casos.Por tanto, dado un punto p = (x,y) de la imagen I, la matriz Hessiana H(p,σ) en el punto p a la escala σ se define como:

$$H(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix} \tag{4.8}$$

Donde $L_{xx}(p,\sigma)$ es la convolución de la derivada parcial de segundo orden de la Gaussiana $\frac{\partial^2}{\partial x^2}g(\sigma)$ con la imagen I en el punto p. Lo mismo ocurre para $L_{xy}(p,\sigma)$ y $L_{yy}(p,\sigma)$.

A pesar de que los filtros Gaussianos son óptimos para el análisis del espacio escala, debido a una serie de limitaciones de estos filtros (como la necesidad de ser discretizados, la no prevención totalmente del efecto aliasing, etc.) se ha probado en el detector SURF con una alternativa a los filtros Gaussianos: los filtros de caja. Éstos aproximan las derivadas parciales de segundo orden de las Gaussianas y pueden ser evaluados de manera muy rápida usando imágenes integrales, independientemente del tamaño de éstas. La escala más pequeña que vamos a tener corresponde a un filtro de caja de dimensión 9x9, correspondiente a las aproximaciones de la derivada parcial de segundo orden de una Gaussiana con σ = 1,2. Las aproximaciones de las derivadas parciales se denotan como D_{xx}, D_{xy} y D_{yy}. En cuanto al determinante de la matriz Hessiana, éste queda definido de la siguiente manera:

$$det(H_{aprox}) = D_{xx}D_{yy} - (0{,}9D_{xy})^2 \tag{4.9}$$

Los espacios de escala son a menudo implementados como pirámides de imágenes, en las que éstas son suavizadas repetidamente con un filtro Gaussiano y posteriormente submuestreadas para alcanzar un nivel más alto en la pirámide. En el detector SURF, debido al uso de filtros de caja e imágenes integrales, no se tiene que aplicar iterativamente el mismo filtro a la salida de una capa filtrada previamente, sino que se pueden aplicar dichos filtros de cualquier tamaño a la misma velocidad directamente en la imagen original. De modo que el espacio escala es analizado por medio de ir elevando el tamaño del filtro, en vez de ir reduciendo el tamaño de la imagen (tal y como se hacía con el detector SIFT). La salida obtenida de aplicar el filtro de dimensión 9x9 es la considerada como escala inicial (s = 1.2, correspondiente a una Gaussiana con σ = 1.2). Las sucesivas capas se van obteniendo como

consecuencia de aplicar gradualmente filtros mayores. En cada nueva octava, el incremento de tamaño del filtro es el resultado de doblar el incremento realizado en la octava anterior:

- Octava inicial: $9x9 \xrightarrow{6} 15x15 \xrightarrow{6} 21x21 \xrightarrow{6} 27x27$
- Siguiente octava: $15x15 \xrightarrow{12} 27x27 \xrightarrow{12} 39x39 \xrightarrow{12} 51x51$
- Siguiente octava: $27x27 \xrightarrow{24} 51x51 \xrightarrow{24} 75x75 \xrightarrow{24} 99x99$
- Y así sucesivamente...

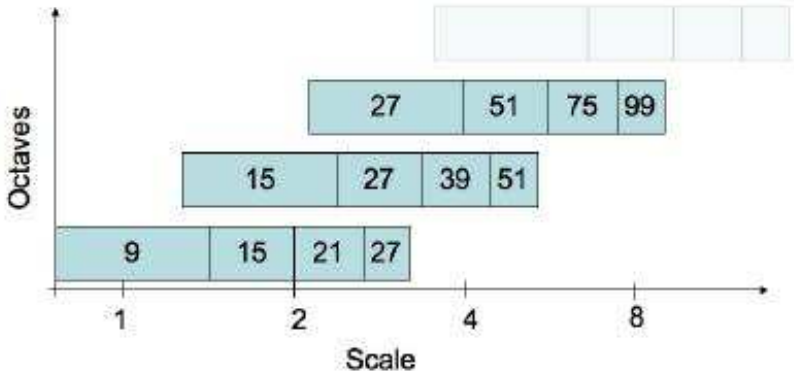


Figura 4.4: Representación gráfica del tamaño de los filtros usados en diferentes octavas. El eje x representa la escala y el eje y indica la octava. Imagen extraída de [1].

Simultáneamente, los intervalos de muestra para la extracción de puntos de interés pueden ser también doblados.

Finalmente, para localizar los puntos de interés en la imagen y en todas las escalas, se procede a la eliminación de los puntos que no sean máximos en una región de vecindad 3x3x3. Entonces, el máximo determinante de la matriz Hessiana es interpolado en la escala y el espacio de la imagen. Con esto ya estaría realizada la detección de los puntos de interés.

4.5.2. Asignación de orientación

El siguiente paso, será la asignación de orientación. Este paso es importante puesto que es la que le otorga al descriptor invarianza ante la rotación, otorgándole a cada punto de interés una orientación. Para ello, lo primero será calcular la Respuesta de Haar en la dirección x e y (ver figura 4.5), en un área circular de vecindad de radio 6s alrededor del punto de interés, donde s es la escala del punto de interés detectado. La etapa de muestreo también depende de la escala y se toma como valor s. En cuanto a las respuestas onduladas de Haar también son calculadas tomando s como referencia, a mayor valor de escala, mayor es la dimensión de las respuestas onduladas. Una vez hecho esto, usamos nuevamente imágenes integrales para un filtrado rápido. Para obtener la respuesta en la dirección x e y se necesitan únicamente 6 operaciones. La longitud de las ondas es 4s.

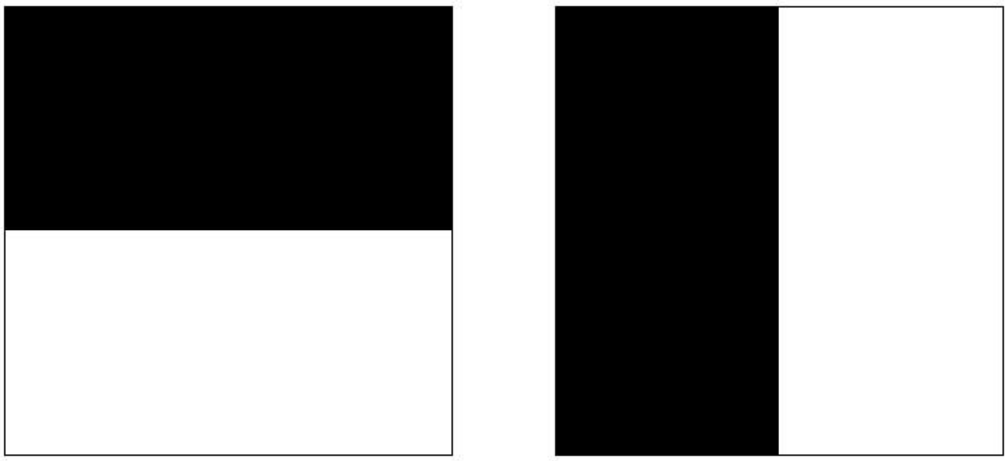


Figura 4.5: Funciones de Haar empleadas en el detector SURF.

Una vez que las respuestas onduladas han sido calculadas, se ponderan con una Gaussiana de $\sigma = 2{,}5s$ centrada en el punto de interés. Las respuestas se representan como vectores en el espacio con la respuesta horizontal a lo largo del eje de abscisas y con la respuesta vertical a lo largo del eje de ordenadas. Para obtener la orientación dominante, se calcula la suma de todas las respuestas dentro de una ventana de orientación móvil (este parámetro se calcula experimentalmente) cubriendo un ángulo de $\frac{\pi}{3}$. Tanto la respuesta vertical como la horizontal dentro de la ventana son sumadas, formando las sumas resultantes un nuevo vector. El vector de mayor longitud es el que da su orientación al punto de interés.

4.5.3. Extracción del descriptor

Para la extracción del descriptor, lo primero que vamos a hacer es la construcción de una región cuadrada alrededor del punto de interés y orientada en relación a la orientación calculada en el paso anterior. El tamaño de la región cuadrada es 20s. Entonces, la región es reducida progresivamente en regiones más pequeñas 4x4. Para cada nueva subregión, se calculan características en puntos de muestra separados por una región 5x5. La respuesta de Haar en la dirección horizontal es llamada d_x, mientras que en la dirección vertical es llamada d_y (la dirección horizontal y vertical se definen en función de la orientación del punto de interés seleccionado). Para darle mayor robustez ante deformaciones geométricas y errores de posicionamiento, las respuestas d_x y d_y son ponderadas con una Gaussiana de $\sigma = 3{,}3s$ centrada en el punto de interés.

Una vez tenemos esto, las respuestas d_x y d_y se suman en cada subregión y conforman un primer conjunto de entradas para el vector de características. Además, para recoger información de la polaridad de los cambios de intensidad, se realiza la suma de los valores absolutos de las respuestas $|d_x|\,|d_y|$. De modo que cada subregión tiene como descriptor un vector v de dimensión 4 para describir su estructura de intensidad:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{4.10}$$

Esto resultará en un descriptor para las 4x4 subregiones de longitud 64.

4.5.4. *Matching*

La última etapa del detector, consistirá en enlazar los puntos de interés hallados en dos imágenes consecutivas. Cada punto de interés en la imagen 1 (instante t) será comparado con los puntos de interés de la imagen 2 (instante t + 1) por medio del cálculo de la distancia Euclídea entre sus descriptores. De este modo, un emparejamiento es detectado en caso de que la distancia relativa entre ambos puntos sea menor a 0.7 veces la distancia respecto al segundo vecino más cercano. Esta estrategia de emparejamiento se conoce como la del vecino más próximo. Además, añadiendo restricciones geométricas adicionales se disminuye el riesgo de incurrir en falsos emparejamientos.

Finalmente, en el artículo presentado por Bay et al [1], podemos encontrar información acerca del rendimiento obtenido con diferentes implementaciones de este detector.

4.6. SIFT vs. SURF

Para determinar cuál de los dos detectores deberíamos usar en nuestra aplicación, vamos a tener en cuenta, además de nuestra propia evaluación, el estudio [18] realizado por los investigadores Johannes Bauer, Niko Sünderhauf y Peter Protzel (pertenecen al departamento de Ingeniería Eléctrica y Tecnología de Información de la Chemnitz University of Technology) en el año 2007.

4.6.1. Estudio realizado por Bauer et al.

En este estudio, estos investigadores evaluaron el comportamiento de varias implementaciones de los detectores SIFT y SURF. Además, compararon ambos detectores con el detector de esquinas de Harris. El entorno en el que realizaron su estudio fue exterior, puesto que era el ambiente en el que operaba su robot. A pesar de que esto es una diferencia importante respecto a nuestra aplicación (pensada para un robot que actúe en el interior de un edificio), vamos a dar por válidos los resultados de su estudio, puesto que nos servirá como referencia para evaluar características de los detectores tales como la velocidad de cómputo, número de puntos de interés localizados o su robustez ante posibles transformaciones de la imagen.

Las conclusiones a las que llegaron son las siguientes:

- Las implementaciones de SIFT detectan generalmente mayor número de puntos de interés que las implementaciones de SURF.
- La calidad de los puntos de interés y los enlaces realizados entre puntos de diferentes imágenes es muy similar en las implementaciones de ambos detectores, aunque puede considerarse ligeramente superior en las de SIFT.

- El rendimiento del detector de esquinas de Harris es inferior a ambos detectores en todos los sentidos.
- Ambos detectores tuvieron un buen comportamiento ante cambios de rotación, con porcentajes de error inferiores al 5
- Ante cambios de escala de las imágenes, ambos detectores se comportaron de buena manera, aunque a medida que aumentaba el cambio de escala, mayor era la pérdida de calidad en los resultados. El número de enlaces entre los puntos de interés de dos imágenes decrece proporcionalmente con el aumento de la diferencia de escala entre imágenes.
- Ante la introducción de ruido en la imagen, ambos detectores tuvieron buen comportamiento, teniendo una tasa de error inferior al 5
- Los cambios de iluminación son resistidos hasta cierto punto, a partir del cual prácticamente no se detecta ningún punto de interés en la imagen.
- Los cambios en la orientación de la imagen tuvieron una gran repercusión en todas las implementaciones de los detectores, cosa que los autores atribuyen a encontrarse bajo un medio natural

Como apoyo, a continuación se muestran las gráficas con los resultados en los que han basado sus conclusiones:

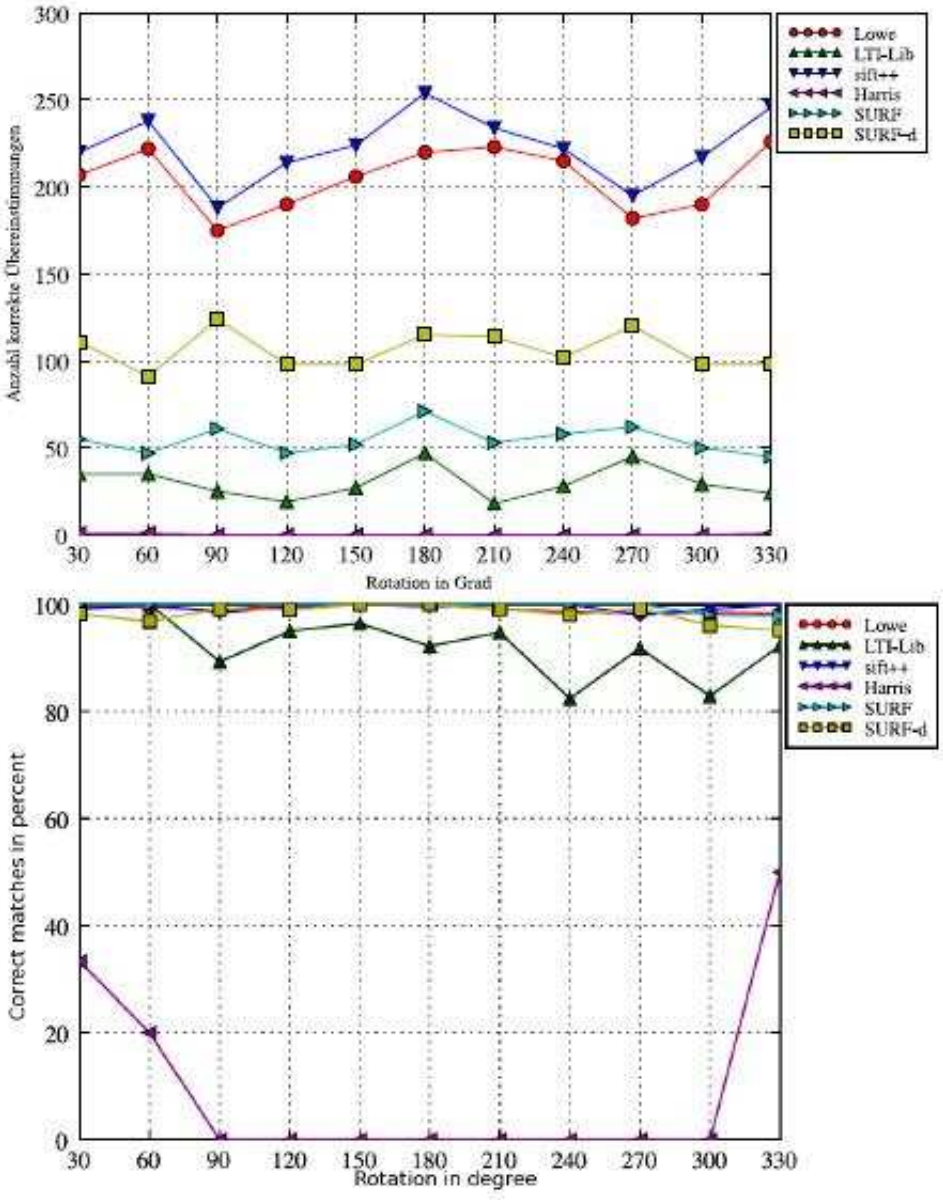


Figura 4.6: Resultados del test de invarianza a la rotación. Imagen extraída de [18]

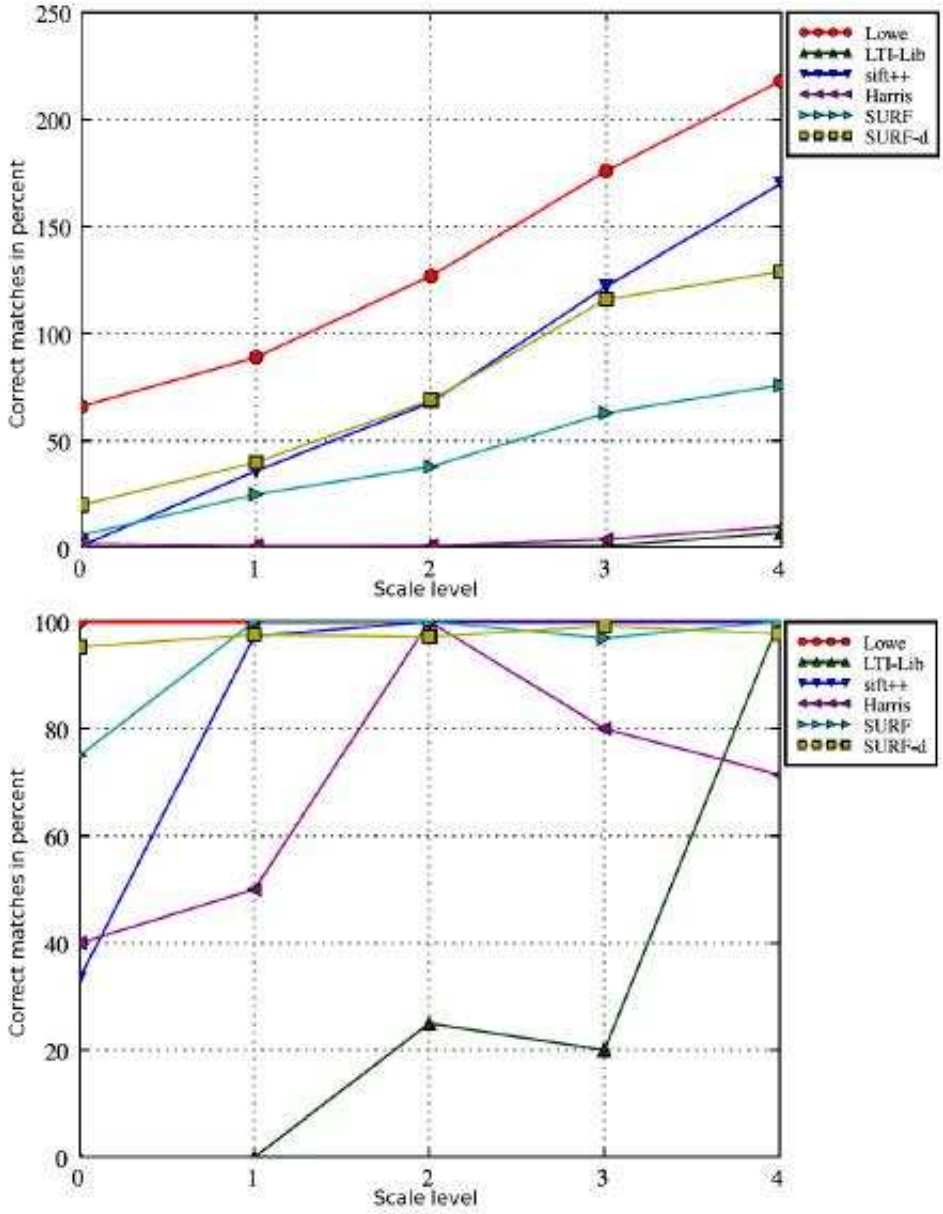


Figura 4.7: Resultados del test de invarianza a la escala. Imagen extraída de [18]

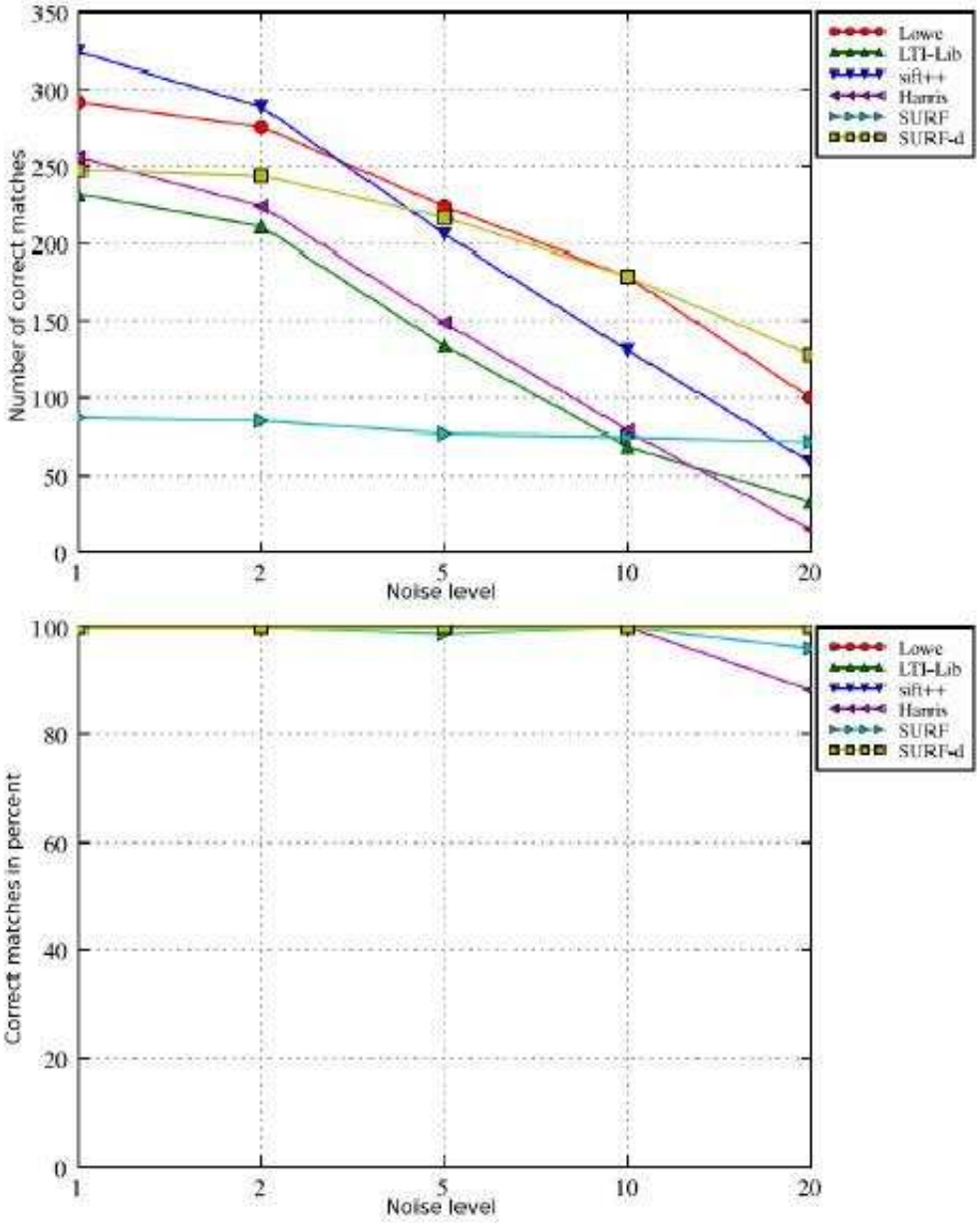


Figura 4.8: Resultados del test de invarianza a la introducción de ruido en la imagen. Imagen extraída de [18]

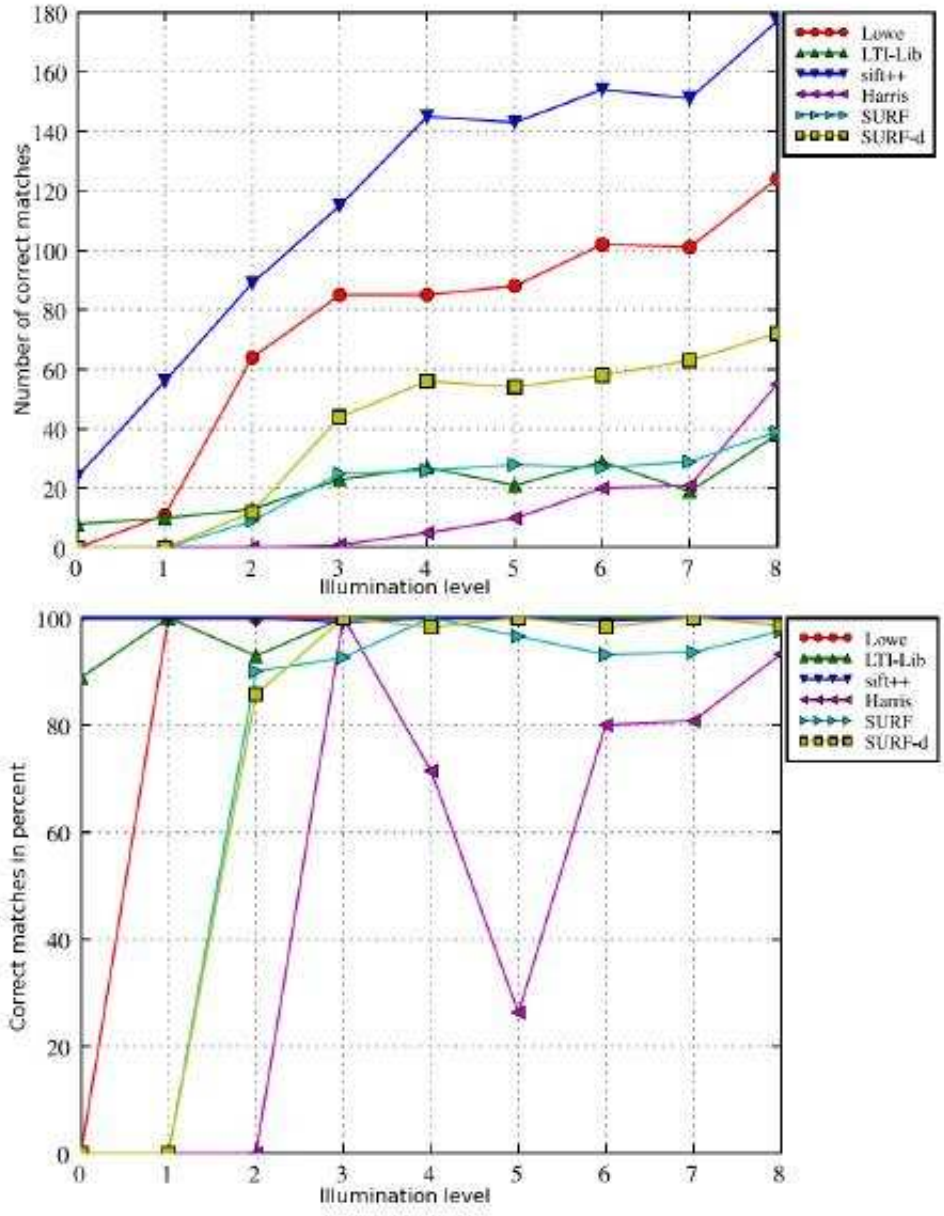


Figura 4.9: Resultados del test de invarianza a cambios de las condiciones de iluminación. Imagen extraída de [18]

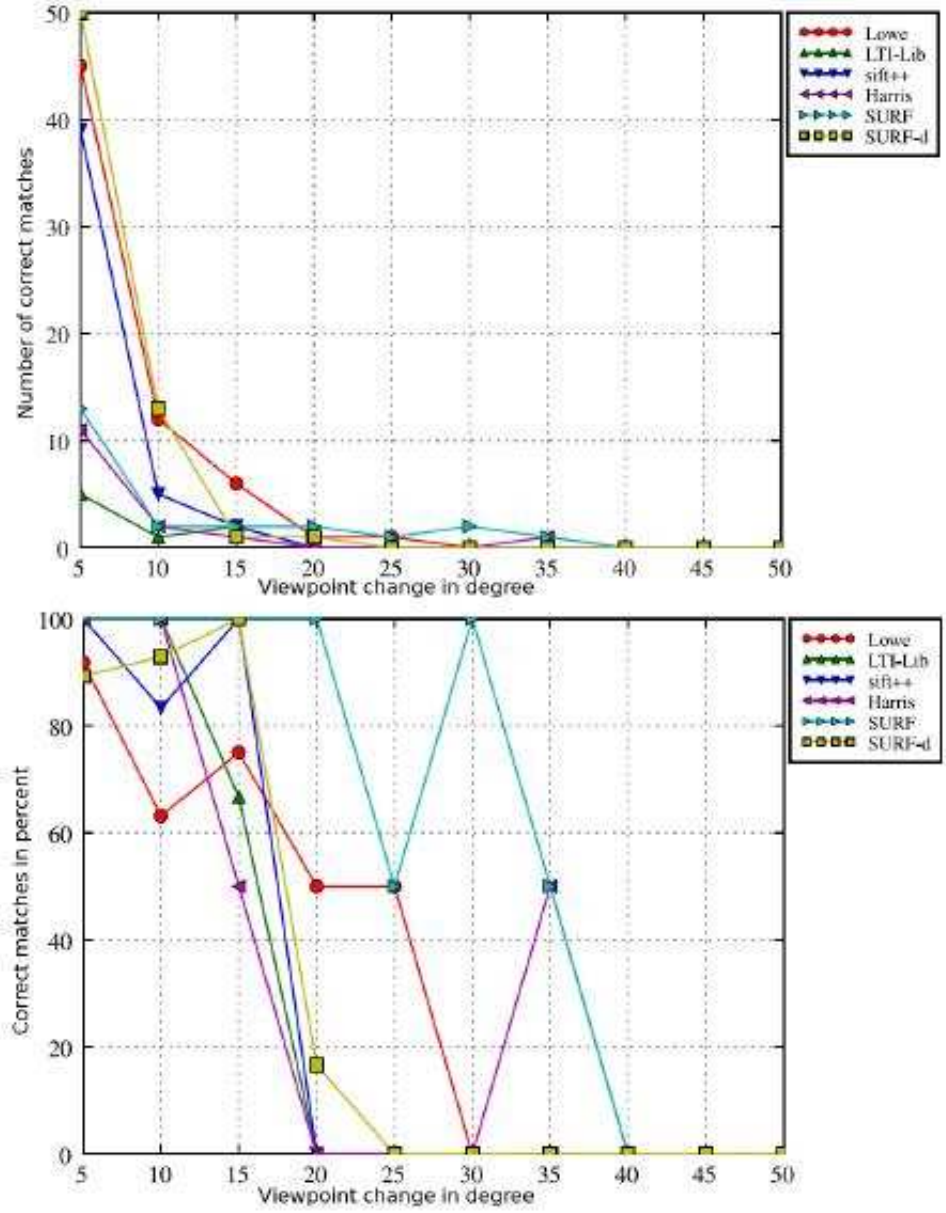


Figura 4.10: Resultados del test de invarianza a cambios en la orientación. Imagen extraída de [18]

Por tanto, a la vista de los resultados de las imágenes, podemos concluir que el detector SURF es superior al SIFT, puesto que los resultados conseguido por ambos son muy similares. El hecho de que el detector SIFT consiga detectar un mayor número de puntos de interés y con menor índice de error, no va a ser para nosotros tan importante, puesto que la diferencia en el resultado es muy pequeña en relación al mayor tiempo de computación que necesita el detector SIFT para conseguir dichos resultados. Además, para nuestra aplicación, no va a ser necesaria la obtención de gran número de puntos de interés, ya que para la determinación de la posición y la distancia recorrida nos bastará con tener al menos un único punto de interés que sea lo suficientemente característico para que se repita en la imagen temporal siguiente.

4.6.2. Estudio propio

Para determinar qué detector debemos usar en nuestra aplicación, vamos a evaluar el rendimiento de las diferentes implementaciones de los detectores SIFT y SURF a las que hemos tenido acceso bajo las siguientes condiciones:

- Utilizaremos diferentes imágenes tomadas del interior de la tercera planta del edificio Bethancourt de la Universidad Carlos III. Este entorno representa perfectamente el ambiente de trabajo de nuestro robot.
- Utilizaremos imágenes tomadas bajo diferentes condiciones de iluminación, orientación, etc.
- Las imágenes utilizadas para la evaluación han sido tomadas con una cámara monocular SONY Handycam DCR-HC19E. A pesar de que nuestro sistema cuenta con una cámara estéreo, la utilización de una cámara monocular para la extracción de imágenes no va a suponer ningún problema.

En cuanto a las diferentes implementaciones que vamos a evaluar, éstas son las siguientes:

- Implementación original del detector SIFT desarrollada por David G. Lowe de la Universidad de British Columbia.

- Implementación en C++ y Matlab del detector SIFT realizada por Andrea Vedaldi de la Universidad de California.
- Implementación en Matlab del detector SIFT de la Universidad de Stanford.
- Implementación original en C y Matlab del detector SURF desarrollada por H. Bay et al.

Los criterios que vamos a valorar para decantarnos por un detector u otro serán los siguientes:

- Rapidez de cómputo.
- Número de errores cometidos.
- Facilidad de acceso al código para poder realizar modificaciones.
- Facilidad de uso de la aplicación.

SIFT: David G. Lowe

Dentro de las implementaciones del detector SIFT, ésta [36] es la que presenta una mayor velocidad de cálculo. Tarda aproximadamente 18 segundos en coger 2 frames de una secuencia de video, almacenar las imágenes, calcular los puntos de interés y ver los puntos de interés que se repiten en ambas imágenes. A continuación, en la figura 4.11 se muestra el resultado de la prueba realizada en el pasillo del edificio Bethancourt:

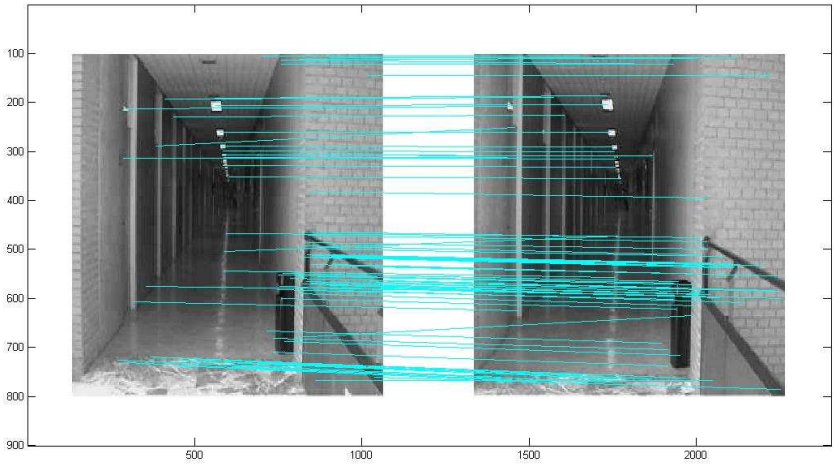


Figura 4.11: Resultado dado por el detector SIFT implementado por David G. Lowe.

Con esta implementación se han detectado 767 puntos de interés en la imagen 1 y 633 puntos de interés en la imagen 2, de los cuáles se han encontrado 93 coincidencias entre ambas imágenes. Si observamos la figura 4.11 podemos ver que el número de errores cometidos al determinar las coincidencias es muy pequeño.

Además del ejemplo mostrado en la figura 4.11, esta implementación se probó para cuatro secuencias de video distintas, representando cada una de ellas un entorno de trabajo en condiciones similares a las que puede encontrarse el robot. A continuación, se muestran los resultados obtenidos con el detector para cada una de esas situaciones:

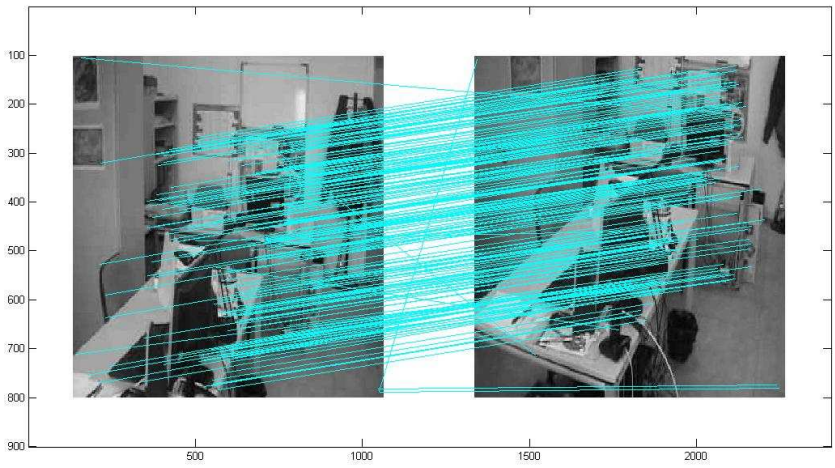


Figura 4.12: Resultado de la implementación en el interior de un despacho.

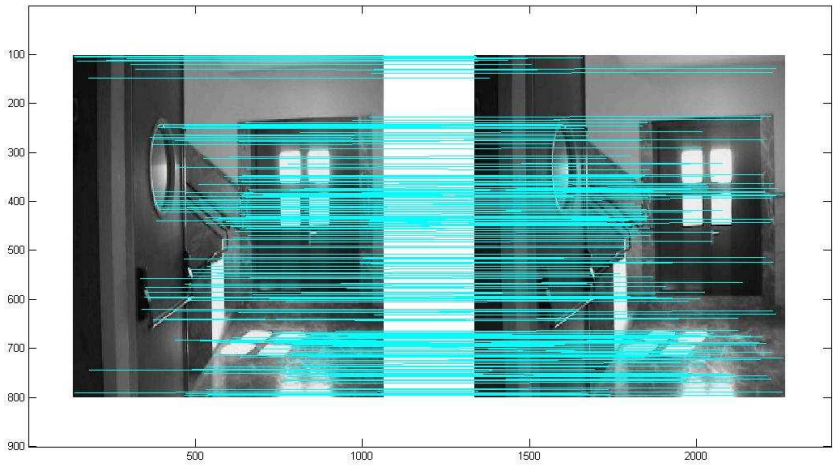


Figura 4.13: Resultado de la implementación en el pasillo interior de un edificio.

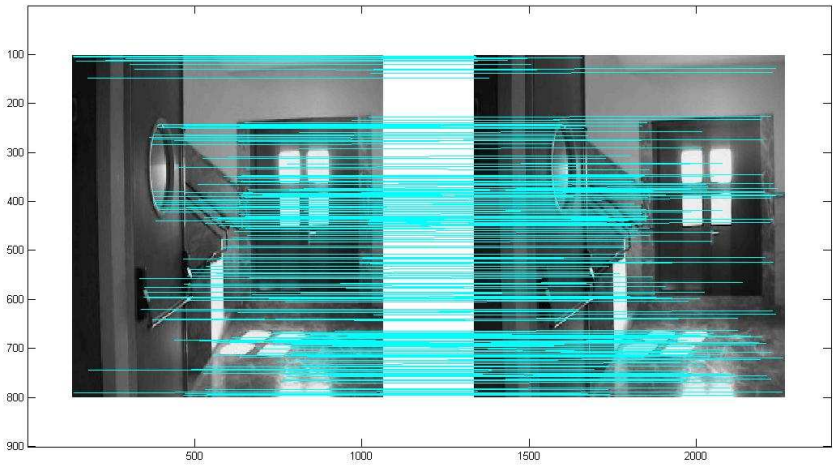


Figura 4.14: Resultado de la implementación en el pasillo interior de un edificio poco iluminado.

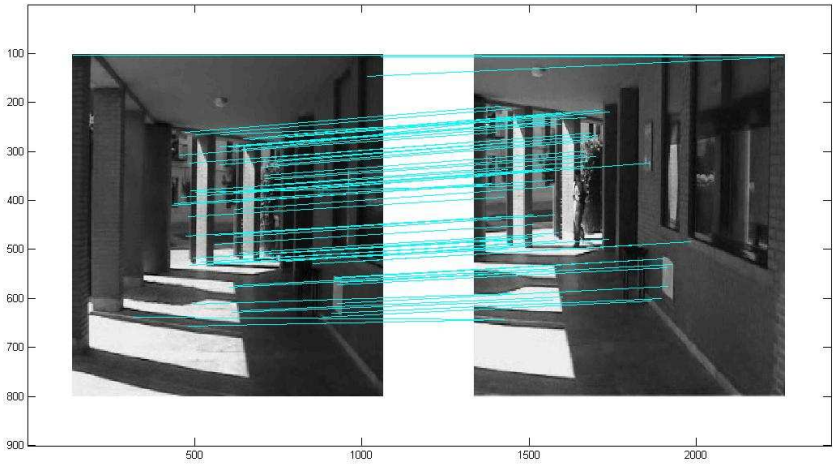


Figura 4.15: Resultado de la implementación en el pasillo exterior de un edificio.

A la vista de los resultados, podemos ver cómo esta implementación tiene un muy buen comportamiento bajo diferentes ambientes de trabajo, ofreciendo una buena respuesta incluso en un ambiente exterior (ver figura 4.18).

En resumen, para este detector tenemos lo siguiente:

Puntos fuertes:

- Pocos errores, buen comportamiento en todas las situaciones evaluadas.
- La más rápida de las implementaciones SIFT.
- Muchos puntos de interés detectados.
- Puede ejecutarse tanto en Windows como en Linux.

Puntos débiles:

- Más lento que el detector SURF.
- Aparentemente comete más errores que SURF.
- Código cerrado, no es posible realizar modificaciones en el algoritmo.

SIFT: Andrea Vedaldi

Esta implementación [35] ha sido desarrollada por el profesor e investigador Andrea Vedaldi de la Universidad de California. Esta versión del detector SIFT es más lenta que la original realizada por David G. Lowe, ya que tarda aproximadamente 1 minuto en coger 2 frames de una secuencia de video, almacenar las imágenes, calcular los puntos de interés y ver los puntos de interés que se repiten en ambas imágenes. A continuación, en la figura 4.16 se muestra el resultado de la prueba realizada en el pasillo del edificio Bethancourt:

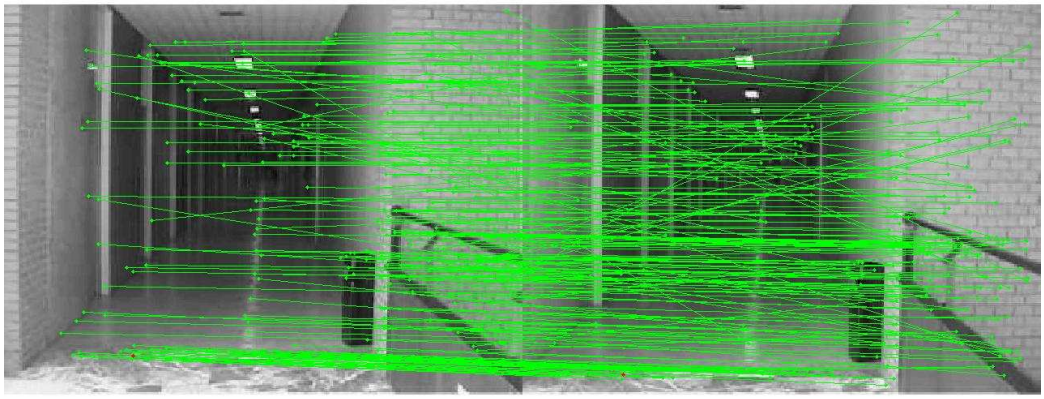


Figura 4.16: Resultado dado por el detector SIFT implementado por Andrea Vedaldi.

Tal y como podemos ver en la imagen, el número de coincidencias encontradas es mayor que con el detector de Lowe. Además, el número de puntos de interés detectados también ha sido mayor, 1015 puntos para la imagen 1 y 749 para la imagen 2. Sin embargo, con este detector tenemos que el número de errores cometidos en la etapa de "matching" es visiblemente mayor. Si tenemos en cuenta el incremento del número de errores y el mayor tiempo de computación necesario con este detector, podemos concluir que este detector no nos interesa para nuestra aplicación, ya que la mejora que presenta frente al algoritmo de Lowe, un mayor número de puntos de interés, no es relevante para nuestra aplicación, ya que, como bien hemos dicho antes, nos basta con unos pocos puntos de interés para realizar los cálculos. La velocidad de cómputo puede reducirse en gran medida si reducimos las dimensiones de las imágenes empleadas en el detector, pero esto conlleva una pérdida considerable tanto en número como en calidad de los puntos de interés detectados. Además, la mejora de velocidad alcanzada no llega a superar al detector de Lowe o al SURF.

A continuación, en las imágenes siguientes se muestran otros resultados alcanzados con este detector en ambientes de trabajo similares a los empleados por nuestra aplicación:

Figura 4.17: Resultado de la implementación en el interior de un despacho.

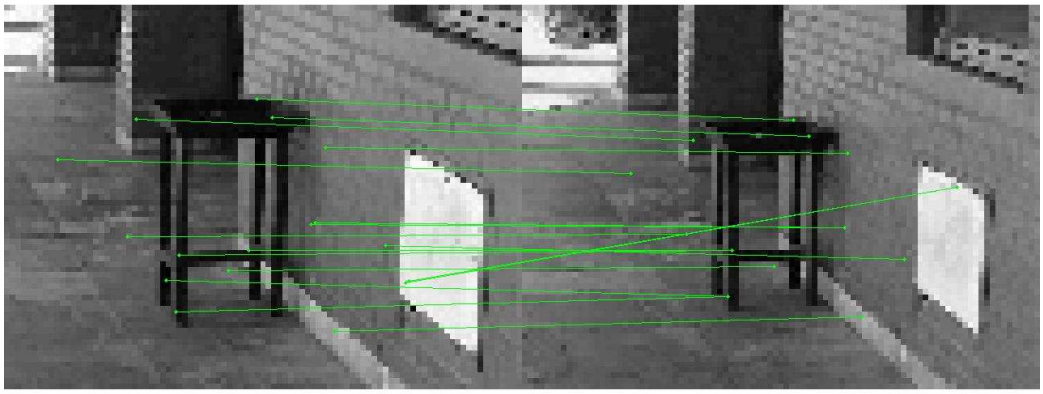


Figura 4.18: Resultado de la implementación en un escenario al aire libre.

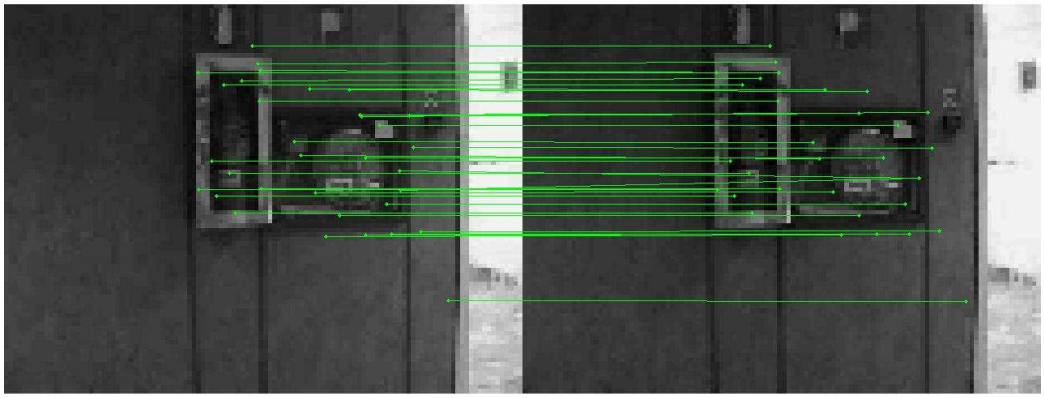


Figura 4.19: Resultado de la implementación en el pasillo interior de un edificio poco iluminado.

En las figuras 4.18 y 4.19 podemos apreciar con claridad la pérdida de calidad sufrida en caso de que reduzcamos el tamaño de la imagen.

En resumen, para este detector tenemos lo siguiente:

Puntos fuertes:

- Código abierto, siendo posible realizar modificaciones en el algoritmo.
- Da una mayor información a la salida.
- Mayor número de puntos de interés detectados.
- Puede ejecutarse tanto en Windows como en Linux.

Puntos débiles:

- Mayor número de errores que la implementación de Lowe y SURF.
- Muy lento a la hora de realizar los cálculos.

SIFT: Universidad de Stanford

Esta implementación fue desarrollada por la Universidad de Stanford [37] y presenta un buen rendimiento en cuanto a velocidad de computación, aunque es inferior a la implementación de David G. Lowe. Con esta implementación tardamos aproximadamente 14 segundos en coger 2 frames de una secuencia de video, almacenar las imágenes y calcular los puntos de interés en ambas imágenes. A diferencia de las implementaciones anteriores, ésta no busca las coincidencias entre los descriptores detectados en ambas imágenes, por lo que, en caso de elegir esta implementación, habría que incluir dicha etapa en el código. A continuación, en las imágenes 4.20 y 4.21 se muestran los resultados obtenidos con este algoritmo para la misma secuencia estudiada en los casos anteriores:

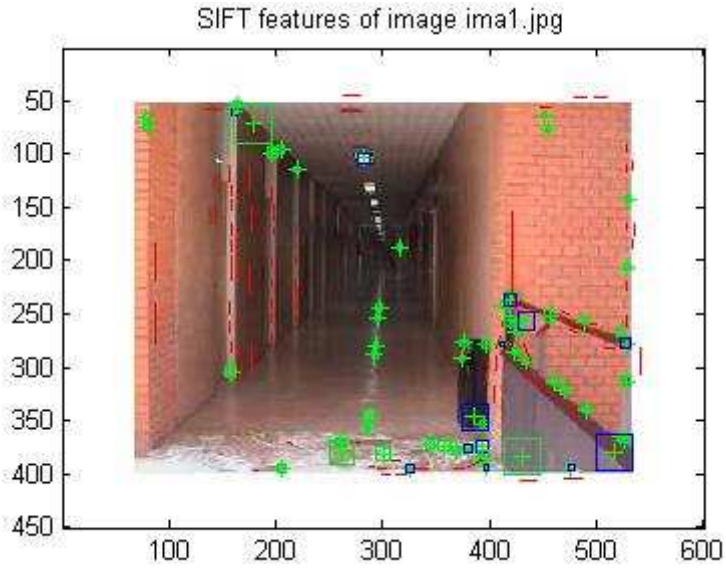


Figura 4.20: Resultado de la implementación para la imagen 1.

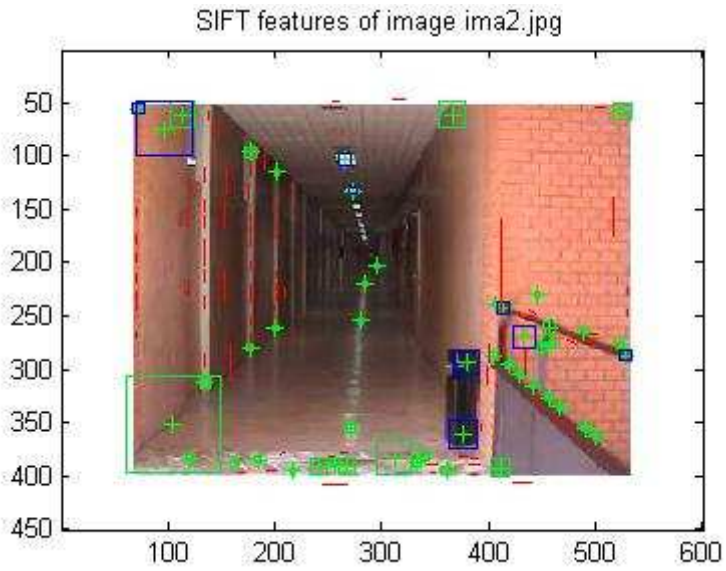


Figura 4.21: Resultado de la implementación para la imagen 2.

Otros resultados obtenidos, bajo condiciones diferentes son los siguientes:

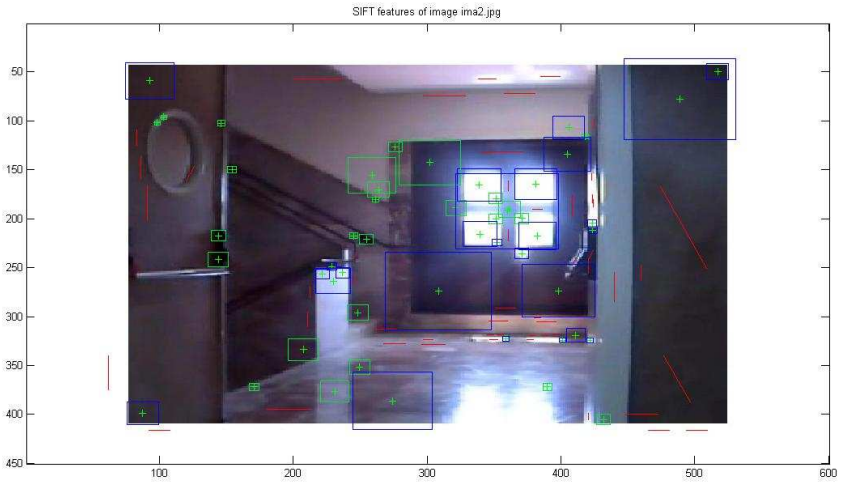


Figura 4.22: Resultado de la implementación en el pasillo interior de un edificio (1).

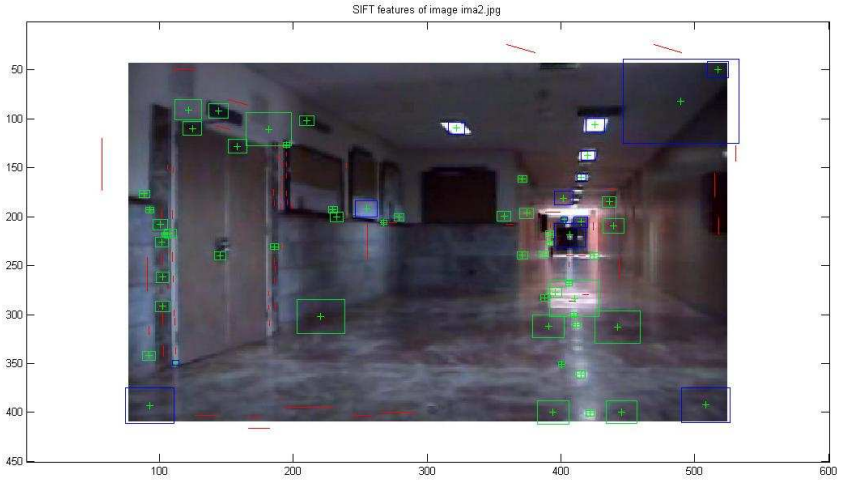


Figura 4.23: Resultado de la implementación en el pasillo interior de un edificio (2).

Figura 4.24: Resultado de la implementación en el pasillo interior de un edificio (3).

En las imágenes podemos ver que el número de descriptores mostrados en la imagen es muy pequeño en comparación con las otras implementaciones. Debido a esto, a no presentar una gran velocidad de computación y al hecho de que no busca las coincidencias entre los descriptores detectados en dos frames consecutivos vamos a desechar esta implementación.

En resumen, para este detector tenemos lo siguiente:

Puntos fuertes:

- Código abierto, siendo posible realizar modificaciones en el algoritmo.
- Puede ejecutarse tanto en Windows como en Linux.

Puntos débiles:

- No realiza etapa de ¨matching¨ entre los descriptores pertenecientes a dos frames consecutivos.
- De todas las implementaciones es la que nos da el menor número de puntos de interés a la salida.

- Para realizar sólo etapa de detección de puntos de interés es muy lento.

SURF: H. Bay

Esta implementación del detector SURF se trata de la desarrollada originalmente a finales del año 2006 por Herbert Bay, Luc Van Gool y Tinne Tuytelaars. En la página web [2], podemos encontrar tanto la implementación del detector para Windows como para Linux, siendo ésta última la que hemos probado. De los detectores estudiados y comparados en esta sección, éste es el que presenta un mejor rendimiento y mayor velocidad de cálculo. Con esta implementación tardamos aproximadamente entre 1 y 1,5 segundos en coger dos imágenes de una secuencia de vídeo capturada por la cámara y calcular los puntos de interés de ambas imágenes. La etapa de "matching" se encuentra separada de la etapa de detección y es llevada a cabo aproximadamente en menos de un segundo. A continuación, se muestra el resultado obtenido con el detector SURF para la misma secuencia de imágenes del pasillo de la tercera planta del edificio Bethancourt con la que hemos probado anteriormente las diferentes implementaciones del detector SIFT:

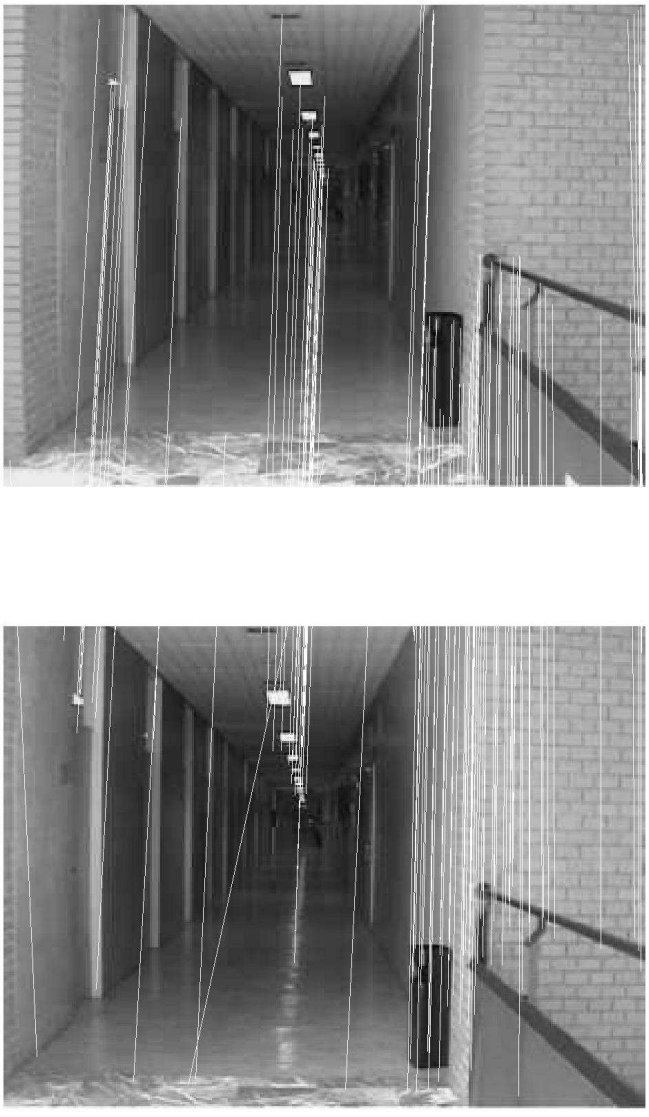


Figura 4.25: Resultado del detector SURF desarrollado por Bay, Van Gool y Tuytelaars.

Tal y como podemos ver en la imagen 4.25, el detector SURF apenas comete errores. Además, el número de emparejamientos realizados es de 107, el cual supone un número mayor a los realizados con el detector SIFT de David Lowe (93). Este dato es interesante ya que el número de puntos de interés detectados por imagen es inferior a los encontrados con las diferentes implementaciones del SIFT, 408 puntos de interés hallados para la imagen 1 y 385 puntos para la imagen 2.

Además de para la secuencia de imágenes que puede verse en la figura 4.25, hemos probado este detector bajo otras condiciones de trabajo para comprobar su rendimiento. Las respuestas obtenidas en las diferentes pruebas pueden apreciarse en las siguientes imágenes:

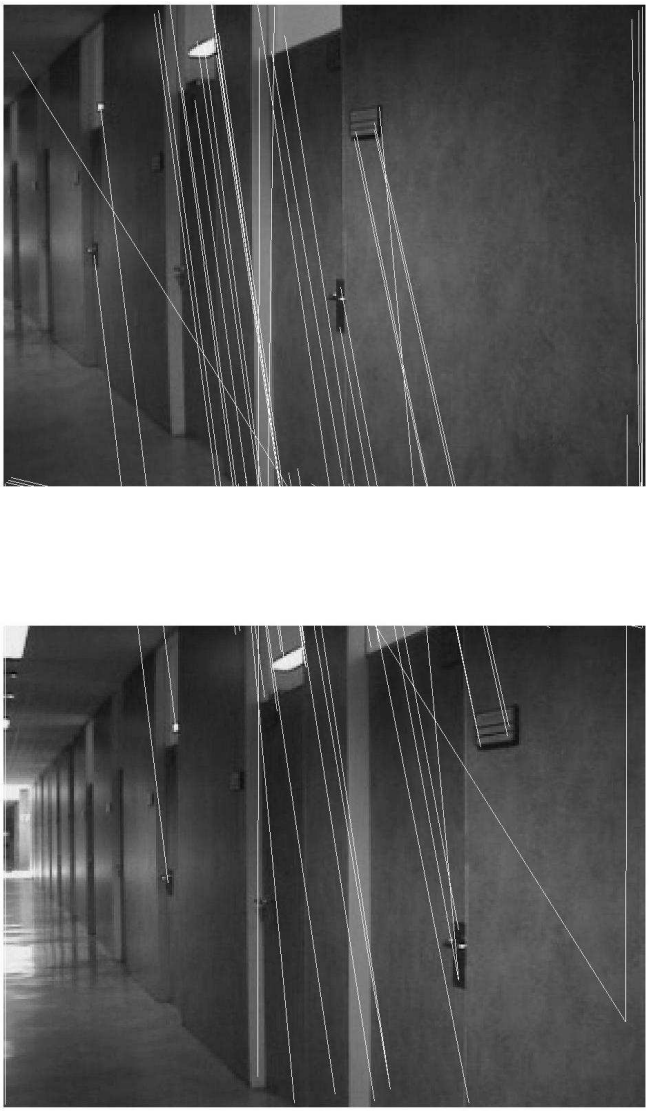


Figura 4.26: Resultado de la implementación en el pasillo interior de un edificio (1).

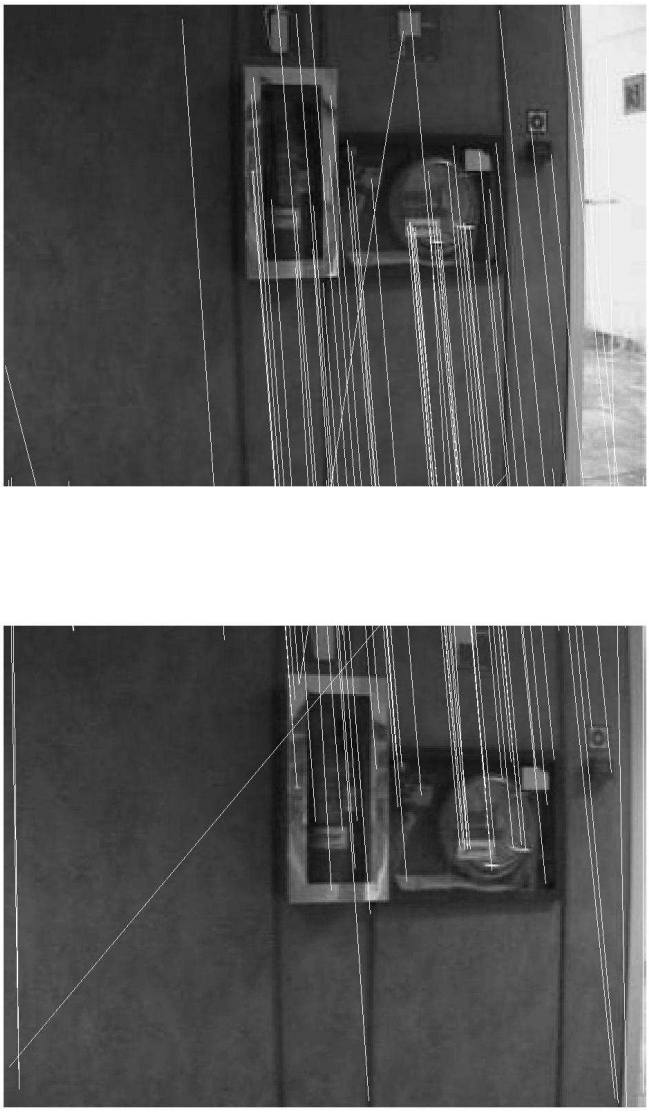


Figura 4.27: Resultado de la implementación en el pasillo interior de un edificio (2).

Figura 4.28: Resultado de la implementación en el pasillo interior de un edificio, bajo condiciones de mucha iluminación.

Figura 4.29: Resultado de la implementación en el interior de un despacho (1).

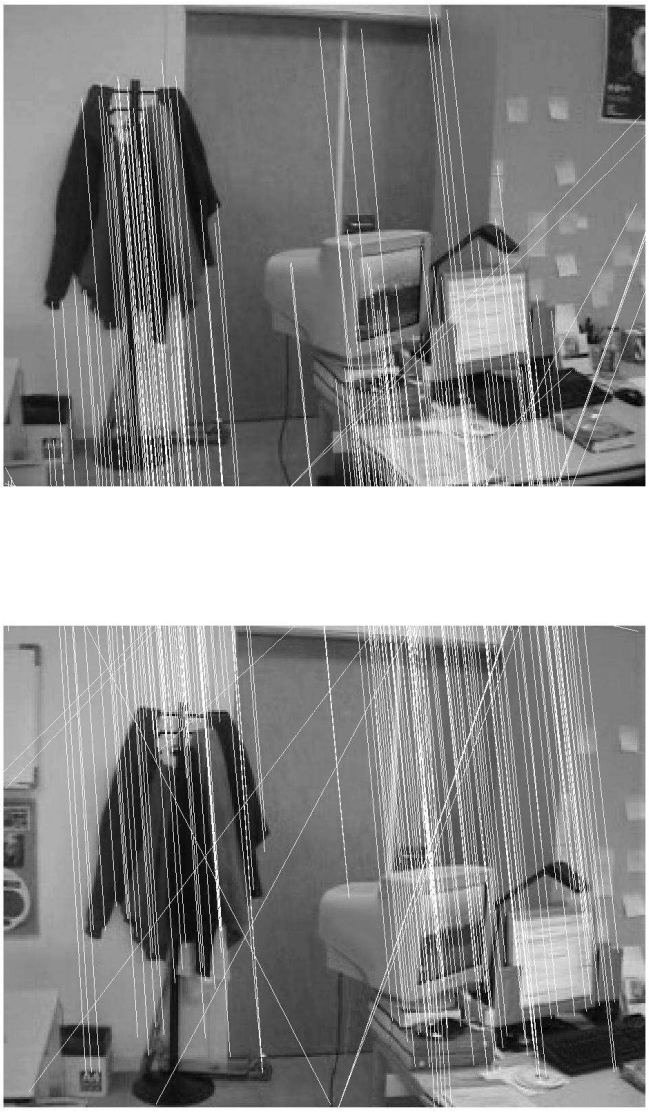


Figura 4.30: Resultado de la implementación en el interior de un despacho (2).

Tal y como podemos ver en los distintos resultados obtenidos, esta implementación es la que posee la mayor velocidad de cálculo de todas las implementaciones y, además, observamos como tiene un muy buen comportamiento bajo distintos ambientes de trabajo, cometiendo un número muy bajo de errores.

En resumen, para este detector tenemos lo siguiente:

Puntos fuertes:

- Comete pocos errores, buen comportamiento en todas las situaciones evaluadas.
- Se trata del detector más rápido, aproximadamente unas diez veces superior al SIFT.
- Puede ejecutarse tanto en Windows como en Linux.
- Código abierto, siendo posible realizar modificaciones en el algoritmo.
- Aporta mucha información a la salida, la mayoría de ella en forma de distintos documentos de texto que son fácilmente accesibles.
- Aparentemente, comete menor número de errores que las implementaciones del detector SIFT.

Puntos débiles:

- Halla menor número de puntos de interés que las implementaciones del detector SIFT.
- Las imágenes de entrada y de salida son dadas bajo formato ".pgm", el cual no es tan común como el formato ".jpg", siendo menos fácil de manejar bajo un entorno Windows.
- La etapa de detección y la de emparejamiento se ejecutan por separado, por lo que habrá que modificar el código para ejecutarlas de manera continua.

Conclusión

Finalmente, si analizamos detenidamente los resultados obtenidos en todas las pruebas, podemos concluir que el detector que mejor se adapta a nuestros intereses es el SURF. La implementación realizada por Bay, Van Gool y Tuytelaars [2] es la que muestra el mejor resultado en cuanto a velocidad de cálculo y su rendimiento es muy bueno, ya que obtenemos un buen número de puntos de interés y el número de errores cometidos en la etapa de emparejamiento es muy bajo. Además, podemos incidir sin ningún tipo de problema sobre su código para adaptarlo mejor a nuestros intereses, siendo posible ejecutarlo tanto bajo un entorno Windows como un entorno Linux.

4.6.3. Otros estudios

Si se quiere profundizar en la comparación de los detectores SIFT y SURF, así como obtener más información acerca de otros posibles detectores, podemos consultar otros documentos de interés como son los estudios realizados por Ballesta et al [14] o el realizado por Tinne Tuytelaars y Krystian Mikolajczyk [20].

CAPÍTULO 5

RESULTADOS

En este capítulo, se va a evaluar el rendimiento del algoritmo implementado en el proyecto. Para ello, vamos a calcular la distancia recorrida y la velocidad llevada por el robot en cuatro situaciones diferentes. Todas las pruebas han sido realizadas en el pasillo de la tercera planta del edificio Bethancourt.

Para el cálculo de la velocidad llevada por el robot, hemos de tener en cuenta el número de frames por segundo que captura la cámara, que para nuestro caso particular es de 11 fps[1]. Para facilitar los cálculos, entre las dos imágenes empleadas en cada situación siempre hay 6 frames, de modo que el tiempo entre ambas imágenes es de 0.545 segundos.

Para llevar a cabo los cálculos, debemos seguir los siguientes pasos:

1. En primer lugar, vamos a separar las imágenes estéreo en dos (imagen de la lente izquierda e imagen de la lente derecha). Para ello, ejecutamos en MATLAB el script *separarimagenes.m* para la imagen capturada en el momento t y la imagen capturada en el momento t +1.

[1] *Frames per second* ≡ *Frames por segundo*

$Ej:\ separarimagenes('Nombreimagen1.jpg','derecha.pgm','izquierda.pgm')$

2. Una vez tenemos separadas ambas imágenes estereoscópicas, para cada una de ellas vamos a obtener sus descriptores surf. Para ello, ejecutamos en la consola de linux la función $surf.ln$

   $Ej:\ ./surf.ln\ -i\ derecha.pgm\ -o\ derecha.surf$

3. Con los descriptores obtenidos para cada imagen, realizamos la etapa de matching para cada par de imágenes estéreo con la función $match.ln$. Adicionalmente, realizamos el emparejamiento de los descriptores obtenidos en las imágenes capturadas por la lente izquierda en ambos instantes. Durante esta etapa, además, se realizará el cálculo de la profundidad de los puntos emparejados.

   $Ej:\ ./match.ln\ -k1\ derecha1.surf\ -k2\ izquierda1.surf\ -im1\ derecha1.pgm\ -im2\ izquierda1.pgm\ -o\ salida.pgm$

4. Una vez obtenidos los emparejamientos, ya podemos calcular la distancia recorrida y la velocidad del robot con el script de MATLAB $distrec.m$.

   $Ej:\ distrec('salida.pgm')$

5. Optimización de los resultados mediante la utilización de herramientas estadísticas.

Todos estos pasos indicados arriba se realizan en aproximadamente 3 segundos, de modo que el tiempo de computación empleado es bastante bajo. Lo cual es muy deseable y, por tanto, será un punto a favor en nuestra aplicación.

Una vez indicados los pasos que vamos a seguir para la obtención de los resultados, se van a mostrar a continuación los resultados obtenidos en cada una de las diferentes situaciones.

Situación 1:

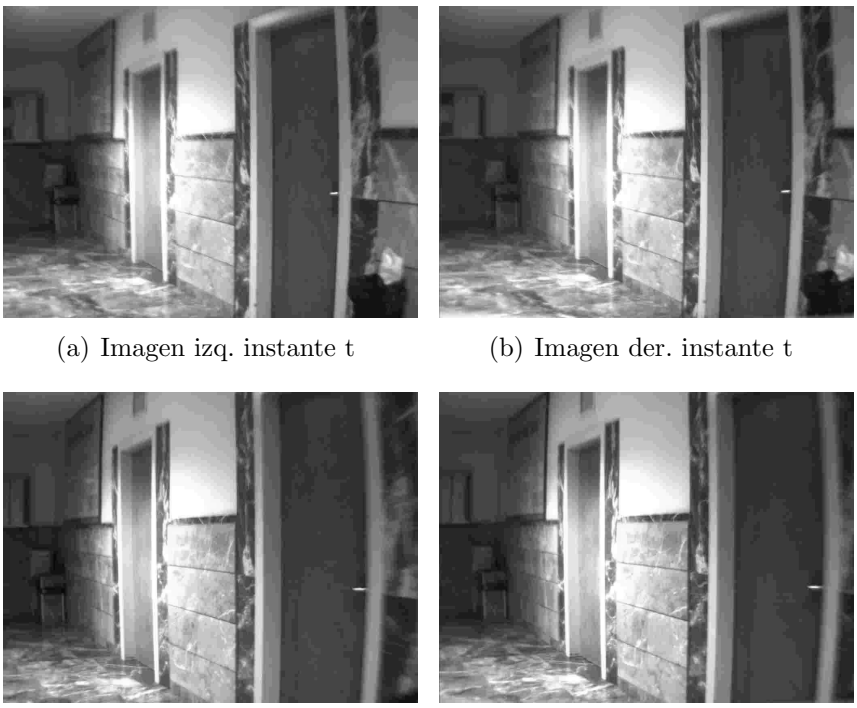
En primer lugar, se muestran las imágenes sobre las que vamos a trabajar en esta situación. En la figura 5.1 podemos ver la imagen estéreo capturada

en el instante t, mientras que en la imagen 5.2 podemos ver la obtenida en el instante t+1.

Figura 5.1: Imagen situación nº1 en el instante t.

Figura 5.2: Imagen situación nº1 en el instante t+1.

A continuación, se muestran las imágenes resultantes de separar ambas capturas estéreo.



(a) Imagen izq. instante t

(b) Imagen der. instante t

(c) Imagen izq. instante t+1

(d) Imagen der. instante t+1

Figura 5.3: Imágenes empleadas en el análisis de la situación 1.

Una vez separadas las imágenes, se realizan las etapas de emparejamiento entre la imagen capturada por la lente izquierda y la lente derecha del instante t y del instante t+1. Los resultados del emparejamiento de imágenes en ambos instantes pueden ver en las imágenes 5.4 y 5.5 respectivamente.

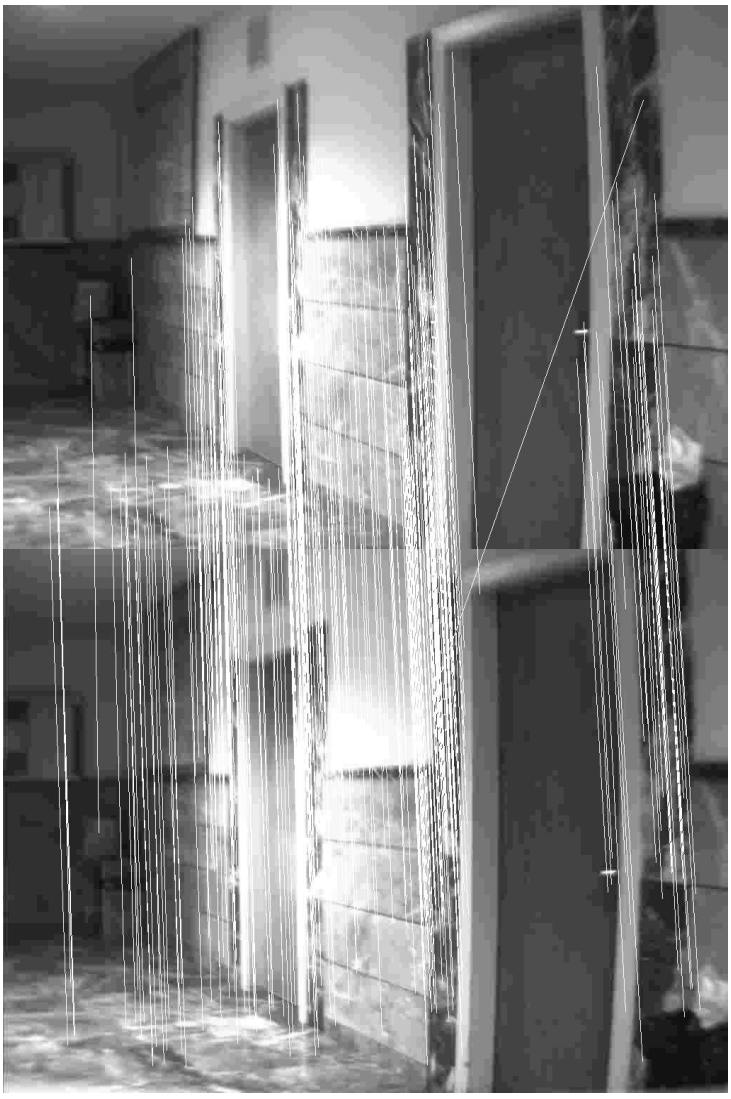


Figura 5.4: Matching entre las imágenes del instante t.

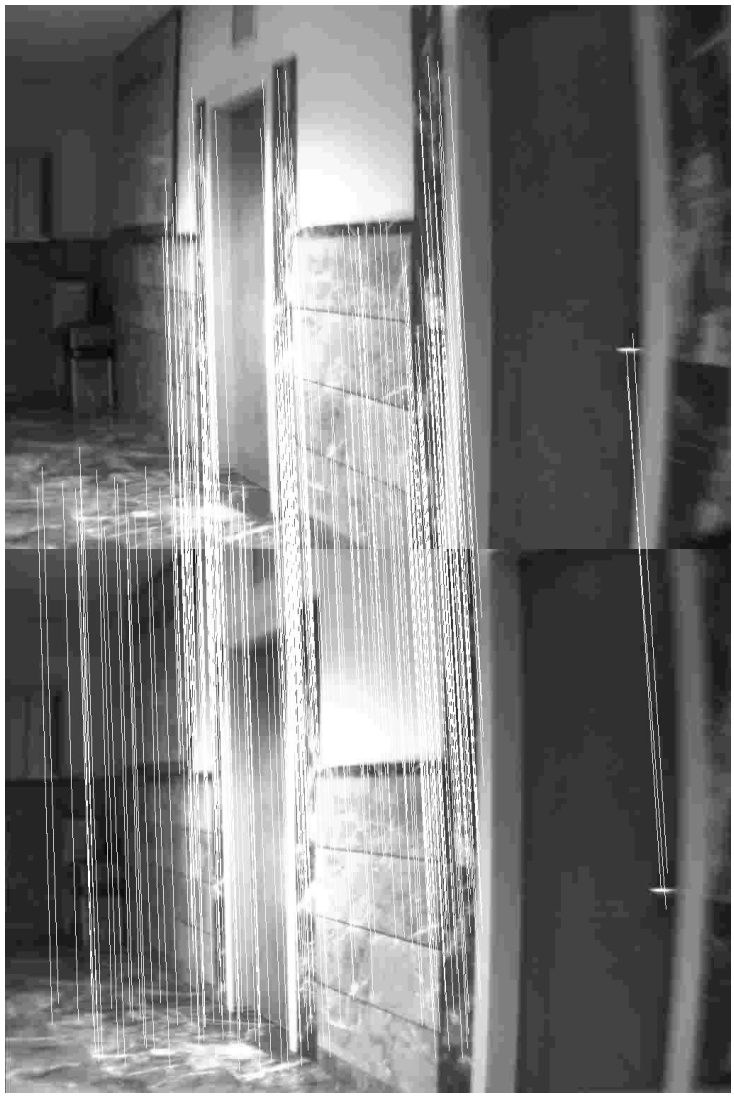


Figura 5.5: Matching entre las imágenes del instante t+1.

Por último, para realizar los cálculos de la distancia recorrida y la velocidad, se ha realizado el emparejamiento de los puntos detectados en la lente izquierda en el instante t y el instante t+1, y a partir de los cuales se ha obtenido el cálculo de la profundidad. Estos puntos pueden verse en la figura 5.6.

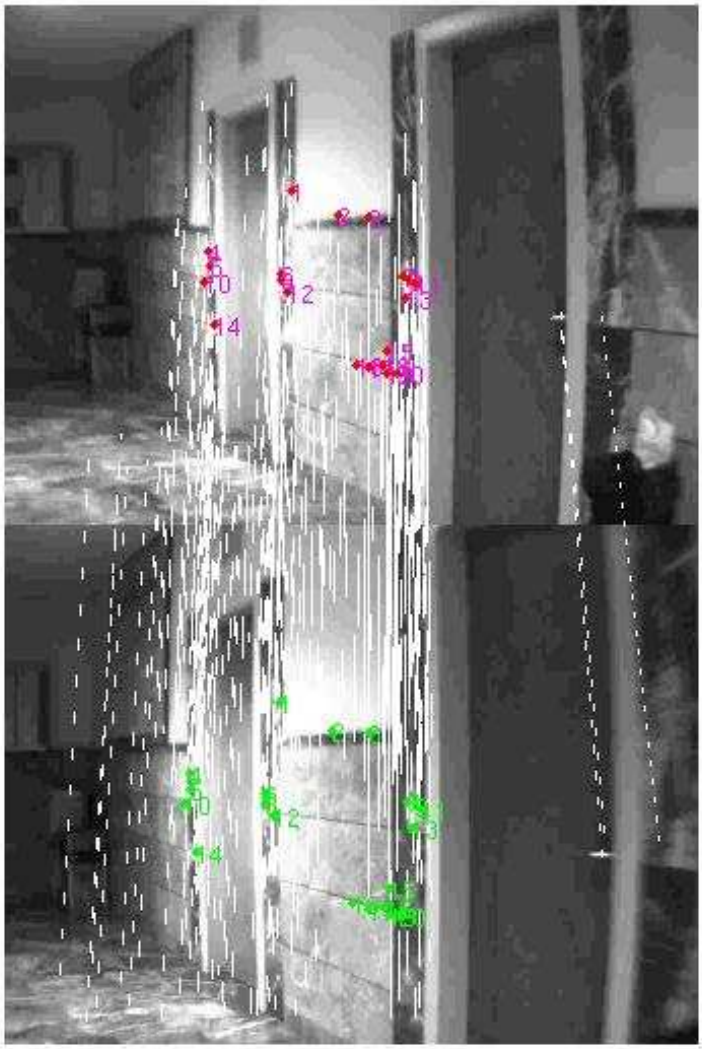


Figura 5.6: Puntos sobre los que se ha obtenido la distancia recorrida en la situación nº1.

En cuanto a los resultados obtenidos con el algoritmo implementado, éstos pueden verse en la siguiente tabla:

Resultados situación 1				
Punto	Profundidad t [m]	Profundidad t+1 [m]	Distancia Recorrida [m]	Desviación de la media [cm]
1	2,9679	2,4289	0,5413	7,748
2	3,0190	2,5241	0,4965	12,228
3	4,0351	3,4049	0,6288	1,002
4	5,6650	4,8524	0,8232	20,442
5	3,3598	2,7380	0,6222	0,342
6	3,5149	2,7676	0,7488	13,002
7	3,6279	2,8548	0,7726	15,382
8	5,3683	4,5660	0,8356	21,682
9	3,1088	2,4549	0,6576	3,882
10	4,0955	3,5508	0,5419	7,688
11	3,2372	2,6047	0,6346	1,582
12	3,1085	2,5119	0,5996	1,918
13	3,0716	2,5141	0,5602	5,858
14	3,0773	2,5691	0,5101	10,868
15	3,1058	2,5538	0,5541	6,468
16	2,6955	2,1974	0,5021	11,668
17	4,1732	3,6633	0,5066	11,218
18	3,2599	2,6224	0,6396	2,082
19	3,1580	2,5900	0,6525	3,372
20	3,0542	2,5094	0,5477	7,108
			Distancia Media Recorrida [m]	0,6188
			Velocidad [m/s]	1,1344

Figura 5.7: Resultados obtenidos en la situación nº1.

Tal y como podemos ver, para el cálculo de la distancia recorrida y la velocidad hemos empleado 20 puntos. El motivo de esto se encuentra en que se busca reducir al máximo el tiempo de computación empleado sin comprometer de manera notable la precisión de los resultados. Por tanto, podemos ver que la distancia media recorrida es de 0.6188 metros y la velocidad llevada por el robot es de 1.13 m/s.

Para comprobar la validez de los puntos empleados, hemos calculado el error producido entre la distancia media obtenida y la distancia obtenida para cada punto concreto. Como podemos ver prácticamente todos los puntos se desvían de la media entre 0 y 12 cm., salvo en cuatro puntos en los que el error es superior. Para ajustar los resultados, vamos a eliminar de manera iterativa aquellos puntos que se alejen más de la media (hemos determinado que no son admisibles aquellos puntos con errores superiores a 15 cm.). Con esto nos quedarían los siguientes resultados:

Resultados situación 1			
Punto	Iteración 1	Iteración 2	Iteración 3
	Desviación de la media [cm]	Desviación de la media [cm]	Desviación de la media [cm]
1	7,748	4,3653	3,3413
2	12,228	8,8453	7,8213
3	1,002	4,3847	5,4088
4	20,442		
5	0,342	3,7247	4,7487
6	13,002	16,3847	
7	15,382		
8	21,682		
9	3,882	7,2647	8,2888
10	7,688	4,3053	3,2813
11	1,582	4,9647	5,9888
12	1,918	1,4647	2,4888
13	5,858	2,4753	1,4513
14	10,868	7,4853	6,4613
15	6,468	3,0853	2,0613
16	11,668	8,2853	7,2613
17	11,218	7,8353	6,8113
18	2,082	5,4647	6,4888
19	3,372	6,7547	7,7788
20	7,108	3,7253	2,7013
Distancia Media Recorrida [m]	0,6188	0,5850	0,5747
Velocidad [m/s]	1,1344	1,0724	1,0536
Tiempo entre imágenes[s]	0,5455		
Desviación típica [cm]	10,2644	6,8085	5,6058
Dispersión [-]	16,59%	11,64%	9,75%

Figura 5.8: Resultados optimizados para la situación nº1.

En la imagen 5.8 podemos ver marcados en rojo aquellos puntos que se han eliminado en cada iteración. Como vemos, se han realizado 3 iteraciones, parándose cuando los valores de las desviaciones se encontraban entre 0 y 8 cm. Además, se incluye el nuevo valor de distancia recorrida, velocidad, desviación típica y dispersión obtenido en cada iteración. Como resultado final vemos que se han eliminado cuatro de los puntos, siendo la distancia media recorrida de 0.5747 metros y la velocidad de 1.0536 m/s. El valor de dispersión final de la muestra es de 9,75 %, lo que puede calificarse como un resultado más que aceptable.

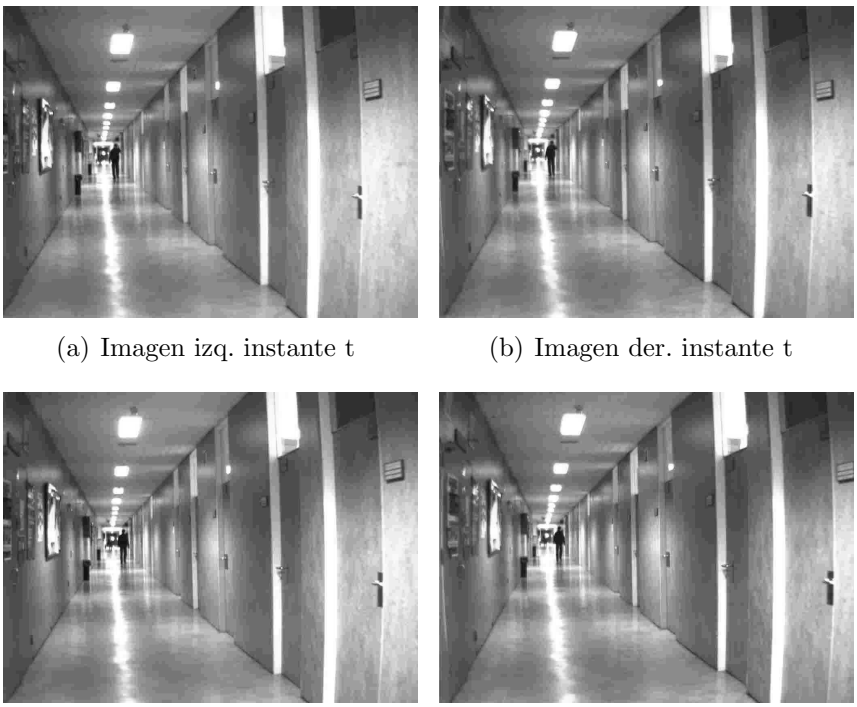
Situación 2:

En primer lugar, se muestran las imágenes sobre las que vamos a trabajar en esta situación. En la figura 5.9 podemos ver la imagen estéreo capturada en el instante t, mientras que en la imagen 5.10 podemos ver la obtenida en el instante t+1.

Figura 5.9: Imagen situación nº2 en el instante t.

Figura 5.10: Imagen situación nº2 en el instante t+1.

A continuación, se muestran las imágenes resultantes de separar ambas capturas estéreo.



(a) Imagen izq. instante t (b) Imagen der. instante t

(c) Imagen izq. instante t+1 (d) Imagen der. instante t+1

Figura 5.11: Imágenes empleadas en el análisis de la situación 2.

Una vez separadas las imágenes, se realizan las etapas de emparejamiento entre la imagen capturada por la lente izquierda y la lente derecha del instante t y del instante t+1. Los resultados del emparejamiento de imágenes en ambos instantes pueden ver en las imágenes 5.12 y 5.13 respectivamente.

Figura 5.12: Matching entre las imágenes del instante t.

Figura 5.13: Matching entre las imágenes del instante t+1.

Por último, para realizar los cálculos de la distancia recorrida y la velocidad, se ha realizado el emparejamiento de los puntos detectados en la lente izquierda en el instante t y el instante t+1, y a partir de los cuales se ha obtenido el cálculo de la profundidad. Estos puntos pueden verse en la figura 5.14.

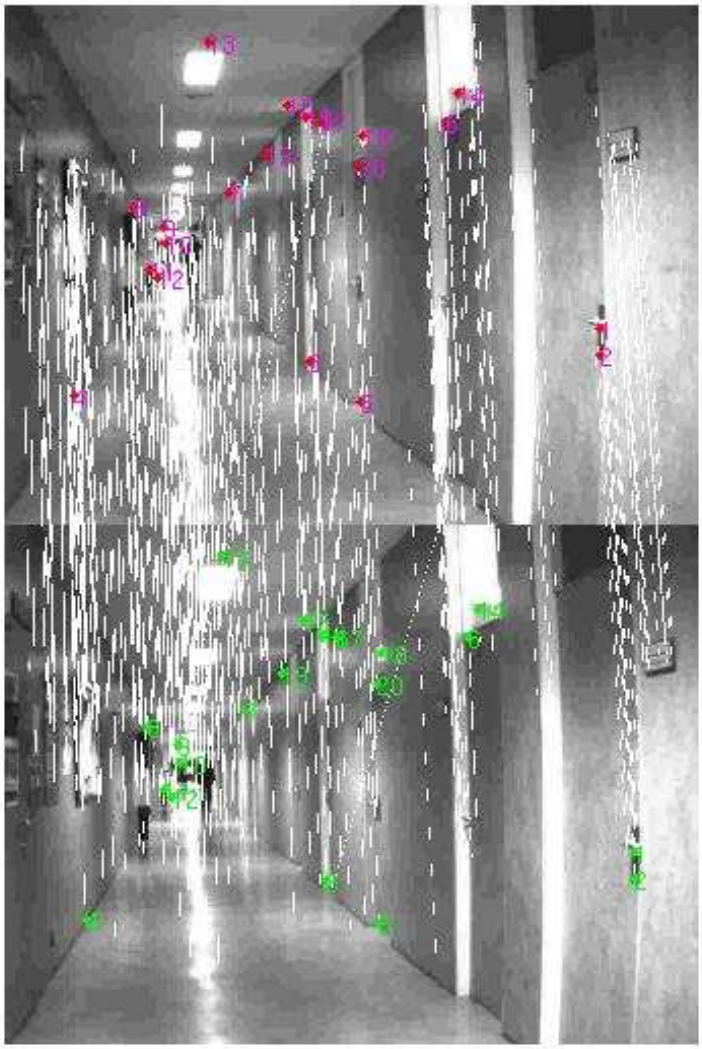


Figura 5.14: Puntos sobre los que se ha obtenido la distancia recorrida en la situación nº2.

En cuanto a los resultados obtenidos con el algoritmo implementado, éstos pueden verse en la siguiente tabla:

Resultados situación 2				
Punto	Profundidad t [m]	Profundidad t+1 [m]	Distancia Recorrida [m]	Desviación de la media [cm]
1	3,1860	3,0874	0,0411	21,76090909
2	3,2263	3,1137	0,0587	20,00090909
3	14,2437	13,5538	0,7125	
4	8,3635	8,6706	0,2963	3,759090909
5	9,0558	8,1110	0,9384	67,96909091
6	4,4605	4,4445	0,0113	24,74090909
7	501,5905	44,0510	467,3702	
8	37,7952	45,4489	8,2716	
9	35,9006	66,9026	34,4170	
10	14,7654	24,1460	10,2568	
11	89,0756	86,3896	3,6097	
12	41,8014	57,1691	16,6201	
13	5,1260	4,9683	0,1893	6,940909091
14	3,8654	3,8182	0,0240	23,47090909
15	12,3567	10,8546	1,5408	
16	9,8704	9,0618	0,8257	56,69909091
17	8,9618	8,9046	0,0661	19,26090909
18	6,9562	6,7395	0,2102	4,850909091
19	14,9952	17,9266	2,9249	
20	7,0253	7,2043	0,1847	7,400909091
			Distancia Media Recorrida [m]	0,2587
			Velocidad [m/s]	0,4743

Figura 5.15: Resultados obtenidos en la situación nº2.

Tal y como podemos ver en la tabla de resultados, en esta situación vemos como producen grandes diferencias a la hora de calcular la distancia recorrida. En primer lugar, debido a que este algoritmo no está diseñado para obtener la distancia recorrida a partir de puntos situados a distancias superiores a los 10 metros, se han descartado todos aquellos puntos situados a profundidades superiores a los 10 metros (estos puntos aparecen señalados en rojo). A pesar de descartar dichos puntos, las desviaciones resultantes en los puntos que nos quedan son tan importantes, que si aplicásemos el mismo procedimiento iterativo para optimizar los resultados, nos quedaríamos con tan sólo tres puntos. Al ser este número de puntos demasiado pequeño, vamos a descartar los resultados obtenidos bajo esta situación.

Por tanto, a la vista de los resultados de esta situación, vamos a descartar la utilización del algoritmo implementado en aquellas situaciones en las que los puntos estén localizados en posiciones muy alejadas.

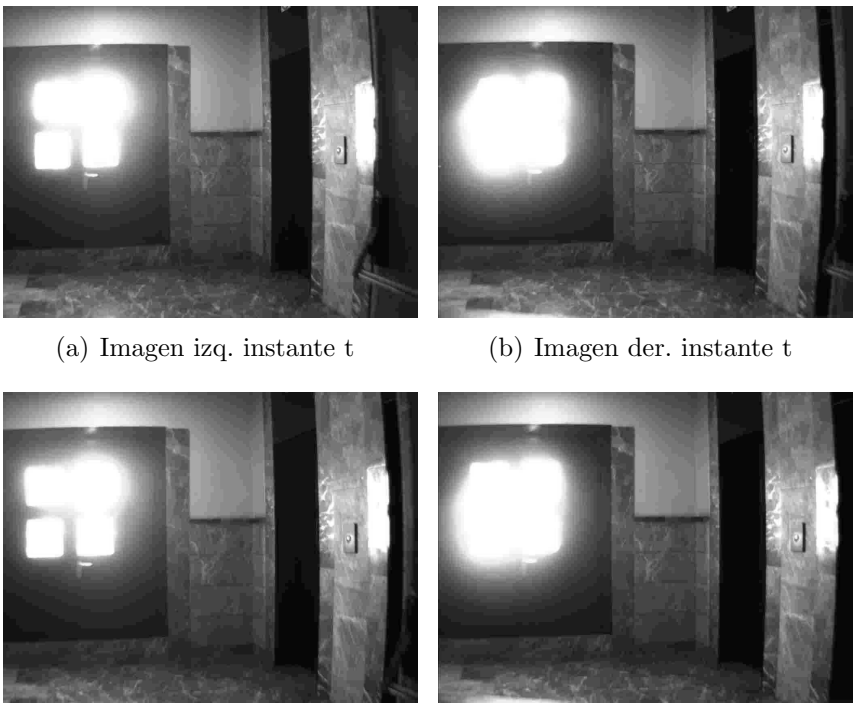
Situación 3:

En primer lugar, se muestran las imágenes sobre las que vamos a trabajar en esta situación. En la figura 5.16 podemos ver la imagen estéreo capturada en el instante t, mientras que en la imagen 5.17 podemos ver la obtenida en el instante t+1.

Figura 5.16: Imagen situación nº3 en el instante t.

Figura 5.17: Imagen situación nº3 en el instante t+1.

A continuación, se muestran las imágenes resultantes de separar ambas capturas estéreo.



(a) Imagen izq. instante t

(b) Imagen der. instante t

(c) Imagen izq. instante t+1

(d) Imagen der. instante t+1

Figura 5.18: Imágenes empleadas en el análisis de la situación 3.

Una vez separadas las imágenes, se realizan las etapas de emparejamiento entre la imagen capturada por la lente izquierda y la lente derecha del instante t y del instante t+1. Los resultados del emparejamiento de imágenes en ambos instantes pueden ver en las imágenes 5.19 y 5.20 respectivamente.

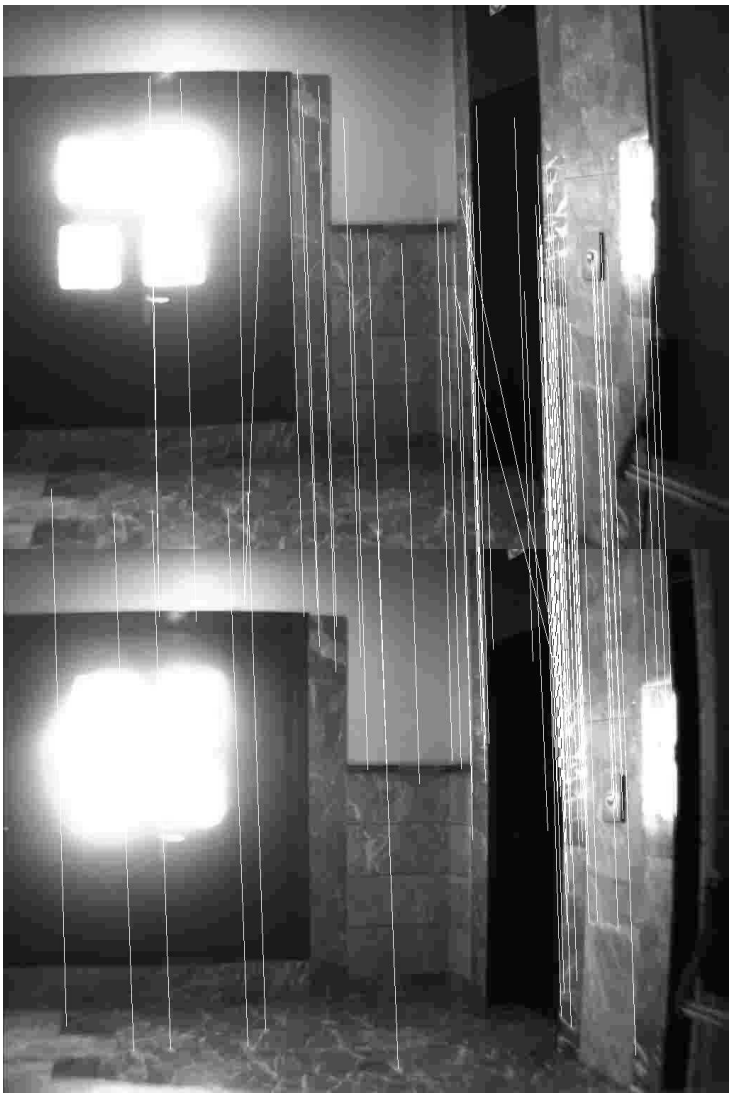


Figura 5.19: Matching entre las imágenes del instante t.

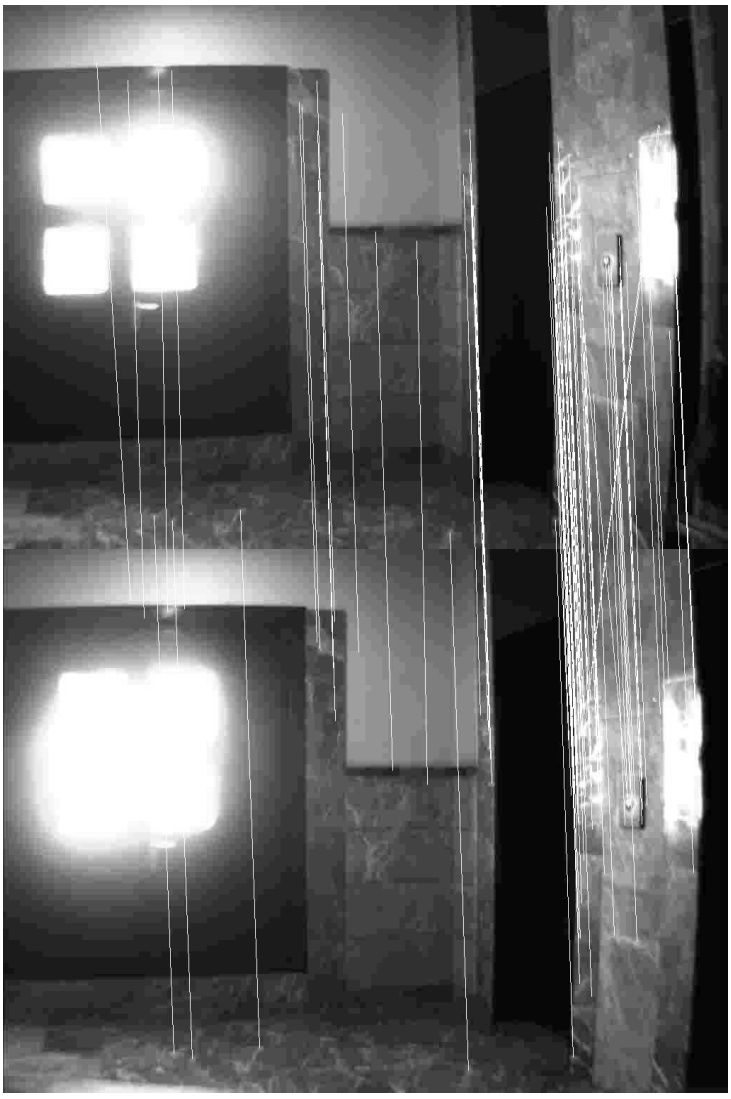


Figura 5.20: Matching entre las imágenes del instante t+1.

Por último, para realizar los cálculos de la distancia recorrida y la velocidad, se ha realizado el emparejamiento de los puntos detectados en la lente izquierda en el instante t y el instante t+1, y a partir de los cuales se ha obtenido el cálculo de la profundidad. Estos puntos pueden verse en la figura 5.21.

Figura 5.21: Puntos sobre los que se ha obtenido la distancia recorrida en la situación nº3.

En cuanto a los resultados obtenidos con el algoritmo implementado, éstos pueden verse en la siguiente tabla:

Resultados situación 3				
Punto	Profundidad t [m]	Profundidad t+1 [m]	Distancia Recorrida [m]	Desviación de la media [cm]
1	3,9673	3,5910	0,3868	12,5435
2	4,6916	4,4554	0,2359	2,5465
3	3,9093	3,8048	0,0902	17,1165
4	3,6309	3,3880	0,2398	2,1565
5	4,8564	4,6207	0,2344	2,6965
6	4,0776	3,7637	0,3178	5,6435
7	4,0487	3,8336	0,2104	5,0965
8	4,1872	3,8760	0,3140	5,2635
9	4,0333	3,8330	0,1944	6,6965
10	3,9546	3,6855	0,2659	0,4535
11	4,0745	3,8849	0,1816	7,9765
12	3,6860	3,4905	0,1850	7,6365
13	3,9563	3,6970	0,2582	0,3165
14	3,9542	3,6592	0,2971	3,5735
15	4,3063	3,9879	0,3218	6,0435
16	3,9272	3,6197	0,3107	4,9335
17	3,3664	3,0603	0,3160	5,4635
18	4,0453	3,8395	0,2012	6,0165
19	3,9960	3,5757	0,4333	17,1935
20	4,7444	4,5103	0,2328	2,8565
	Distancia Media Recorrida [m]		0,2614	
		Velocidad [m/s]	0,4792	

Figura 5.22: Resultados obtenidos en la situación nº3.

Tal y como podemos ver, la distancia media recorrida es de 0.2614 metros y la velocidad llevada por el robot es de 0.4792 m/s.

Observando la diferencia entre la distancia media recorrida y el valor de distancia obtenido en cada punto, podemos ver que todos los puntos se desvían de la media entre 0 y 17 cm., salvo en un caso en el que el error es superior. Para ajustar los resultados, vamos a eliminar aquellos puntos que se alejan más de la media (hemos determinado que no son admisibles aquellos puntos con errores superiores a 10 cm.). Con esto nos quedarían los siguientes resultados:

Resultados situación 3		
Punto	Iteración 1	Iteración 2
	Desviación de la media [cm]	Desviación de la media [cm]
1	12,5435	
2	2,5465	1,8041
3	17,1165	
4	2,1565	1,4141
5	2,6965	1,9541
6	5,6435	6,3859
7	5,0965	4,3541
8	5,2635	6,0059
9	6,6965	5,9541
10	0,4535	1,1959
11	7,9765	7,2341
12	7,6365	6,8941
13	0,3165	0,4259
14	3,5735	4,3159
15	6,0435	6,7859
16	4,9335	5,6759
17	5,4635	6,2059
18	6,0165	5,2741
19	17,1935	
20	2,8565	2,1141
Distancia Media Recorrida [m]	0,2614	0,2539
Velocidad [m/s]	0,4792	0,4655
Tiempo entre imágenes[s]	0,5455	
Desviación típica [cm]	7,6315	4,9079
Dispersión [-]	0,2920	0,1933

Figura 5.23: Resultados optimizados para la situación nº3.

En la imagen 5.23 podemos ver marcados en rojo aquellos puntos que se han eliminado en cada iteración. Como vemos, se han realizado tan sólo 2 iteraciones, parándose cuando los valores de las desviaciones se encontraban entre 0 y 7 cm. Además, se incluye el nuevo valor de distancia recorrida, velocidad, desviación típica y dispersión obtenido en cada iteración. Como resultado final vemos que se han eliminado tres de los puntos, siendo la distancia media recorrida de 0.2539 metros y la velocidad de 0.4655 m/s.

Situación 4:

A continuación, se muestran las imágenes sobre las que vamos a trabajar en esta situación. En la figura 5.24 podemos ver la imagen estéreo capturada en el instante t, mientras que en la imagen 5.25 podemos ver la obtenida en el instante t+1.

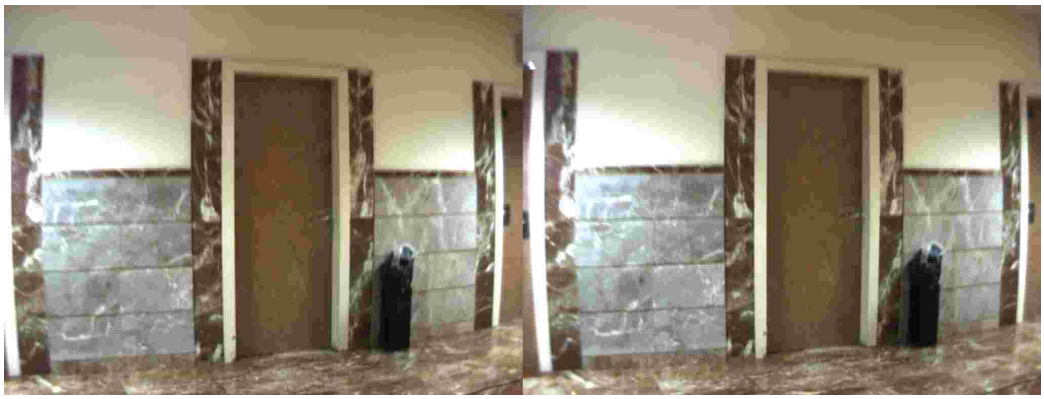
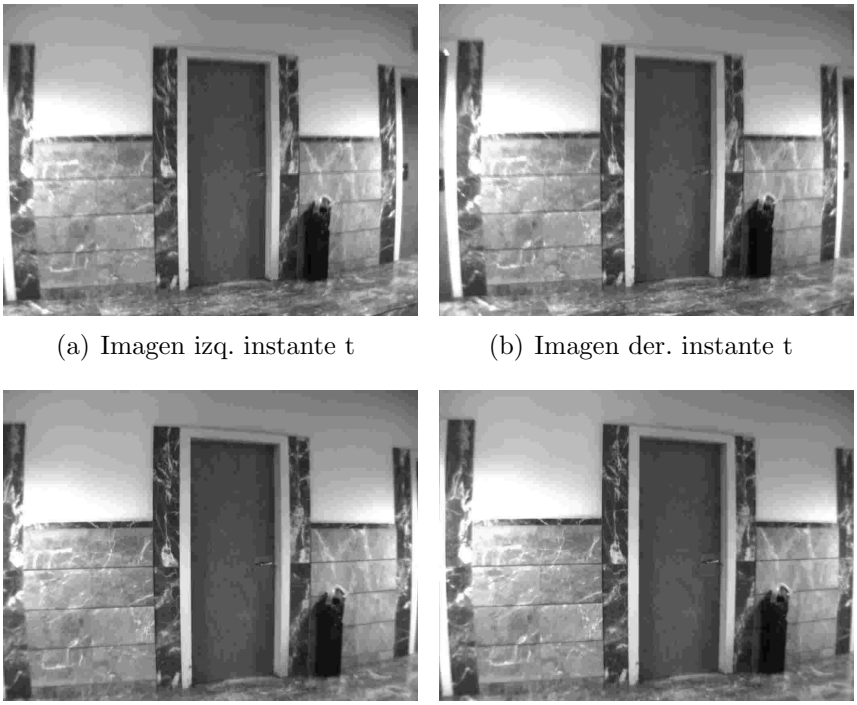


Figura 5.24: Imagen situación nº4 en el instante t.

Figura 5.25: Imagen situación nº4 en el instante t+1.

Ahora, se muestran las imágenes resultantes de separar ambas capturas estéreo.



(a) Imagen izq. instante t

(b) Imagen der. instante t

(c) Imagen izq. instante t+1

(d) Imagen der. instante t+1

Figura 5.26: Imágenes empleadas en el análisis de la situación 4.

Una vez separadas las imágenes, se realizan las etapas de emparejamiento entre la imagen capturada por la lente izquierda y la lente derecha del instante t y del instante t+1. Los resultados del emparejamiento de imágenes en ambos instantes pueden ver en las imágenes 5.27 y 5.28 respectivamente.

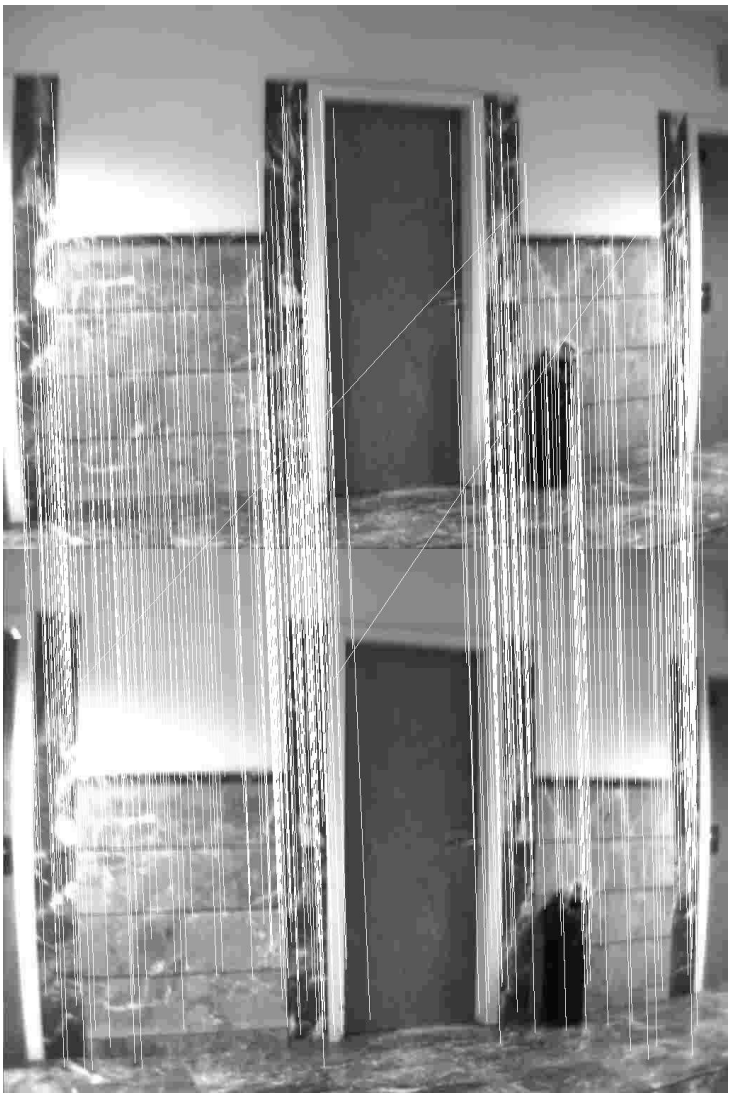


Figura 5.27: Matching entre las imágenes del instante t.

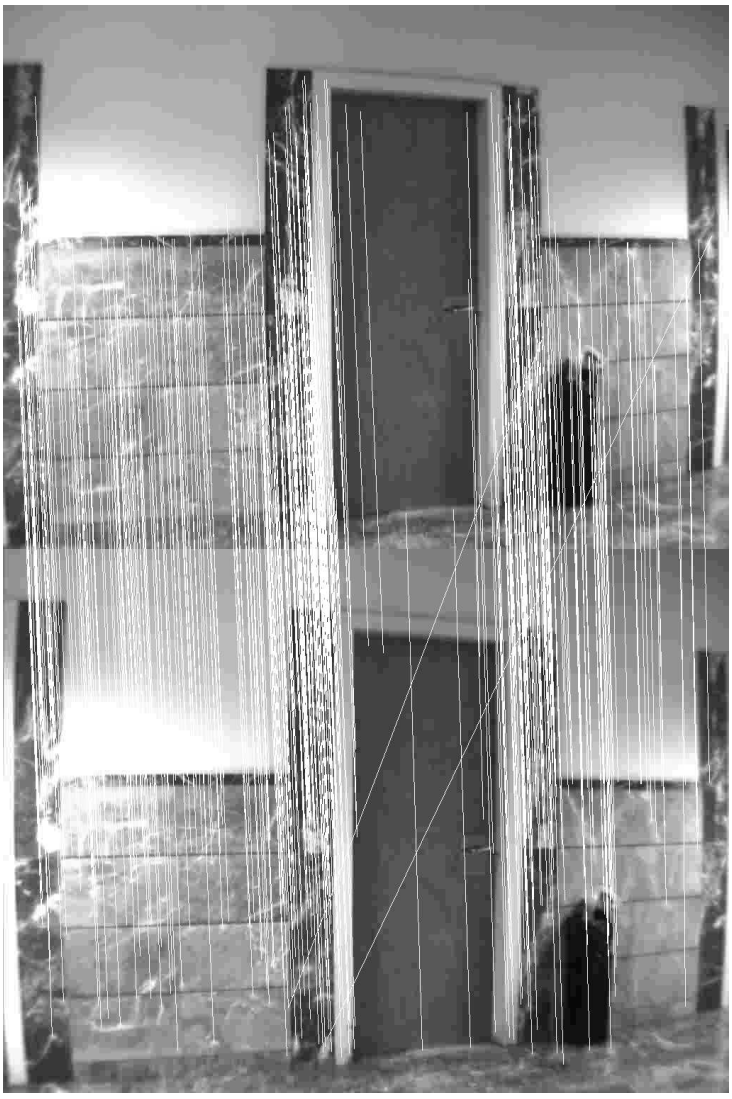


Figura 5.28: Matching entre las imágenes del instante t+1.

Por último, para realizar los cálculos de la distancia recorrida y la velocidad, se ha realizado el emparejamiento de los puntos detectados en la lente izquierda en el instante t y el instante t+1, y a partir de los cuales se ha obtenido el cálculo de la profundidad. Estos puntos pueden verse en la figura 5.29.

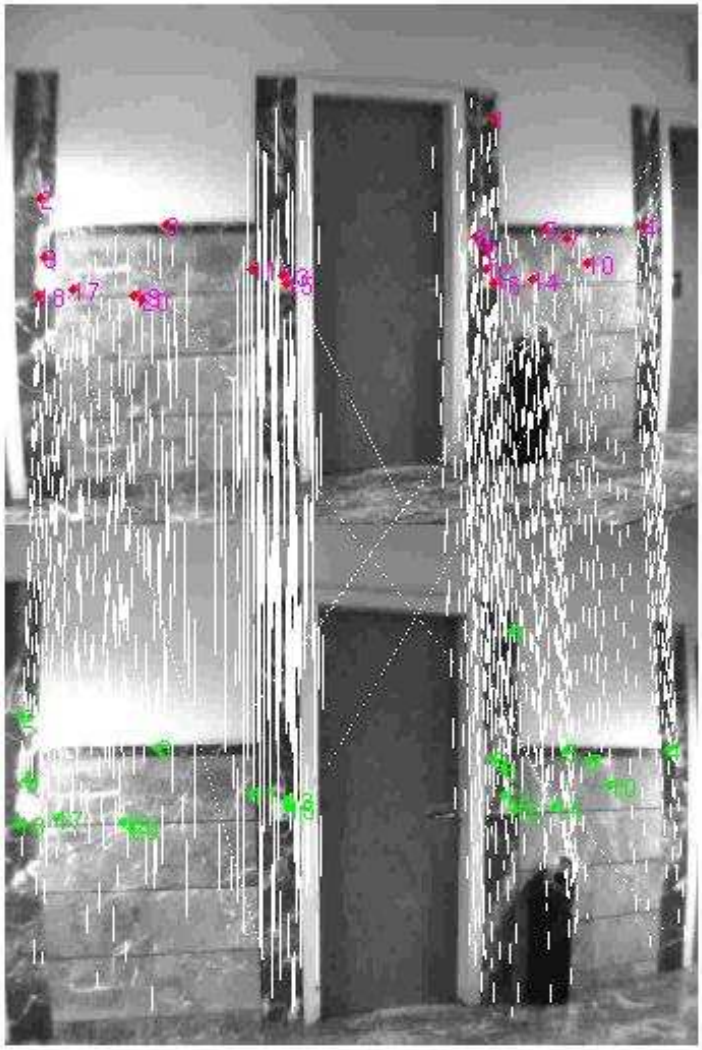


Figura 5.29: Puntos sobre los que se ha obtenido la distancia recorrida en la situación nº4.

En cuanto a los resultados obtenidos con el algoritmo implementado, éstos pueden verse en la siguiente tabla:

Resultados situación 4				
Punto	Profundidad t [m]	Profundidad t+1 [m]	Distancia Recorrida [m]	Desviación de la media [cm]
1	4,9216	4,3676	0,5538	5,5525
2	3,6332	3,3463	0,3096	18,8675
3	3,6262	3,3721	0,2647	23,3575
4	7,5738	7,2408	0,2664	23,1875
5	6,3823	5,4222	0,9973	49,9025
6	4,6767	4,3372	0,3279	17,0375
7	6,4203	5,3321	1,1534	65,5125
8	5,0546	4,5251	0,5266	2,8325
9	3,6949	3,4686	0,2310	26,7275
10	6,7101	5,6428	1,1405	64,2225
11	3,8146	3,4867	0,3368	16,1475
12	5,0869	4,6727	0,4035	9,4775
13	3,9684	3,5167	0,4585	3,9775
14	5,5097	5,3825	0,0818	41,6475
15	4,0906	3,5874	0,5099	1,1625
16	5,1393	4,8119	0,3100	18,8275
17	3,6805	3,3917	0,3090	18,9275
18	3,8453	3,4688	0,4166	8,1675
19	4,0014	3,3434	0,7138	21,5525
20	3,9484	3,3424	0,6544	15,6125
	Distancia Media Recorrida [m]		0,4983	
		Velocidad [m/s]	0,9135	

Figura 5.30: Resultados obtenidos en la situación nº4.

En la imagen 5.30, podemos ver que la distancia media recorrida es de 0.4983 metros y la velocidad llevada por el robot es de 0.9135 m/s.

Analizando las diferencias entre la distancia media recorrida y los valores de distancia obtenidos para cada punto, podemos ver que hay algunos puntos erróneos que hacen que las desviaciones sean muy grandes. Para optimizar los resultados, vamos a eliminar iterativamente aquellos puntos que se alejan más de la media, es decir, progresivamente vamos a ir eliminando aquellos puntos con errores superiores a 30 cm. (1ª iteración), 20 cm. (2ª iteración), 15 cm. (3ªiteración) y 10 cm. (4ª iteración). Haciendo esto nos quedan los siguientes resultados:

Resultados situación 4					
Punto	Solución inicial	Iteración 1	Iteración 2	Iteración 3	Iteración 4
	Desviación de la media [cm]	Desviación de la media [cm]	Desviación de la media [cm]	Desviación de la media [cm]	Desviación de la media [cm]
1	5,5525	14,1769	18,0636		
2	18,8675	10,2431	6,3564	3,5725	1,7567
3	23,3575	14,7331	10,8464	8,0625	6,2467
4	23,1875	14,5631	10,6764	7,8925	6,0767
5	49,9025				
6	17,0375	8,4131	4,5264	1,7425	0,0733
7	65,5125				
8	2,8325	11,4569	15,3436		
9	26,7275	18,1031	14,2164	11,4325	
10	64,2225				
11	16,1475	7,5231	3,6364	0,8525	0,9633
12	9,4775	0,8531	3,0336	5,8175	7,6333
13	3,9775	4,6469	8,5336	11,3175	
14	41,6475				
15	1,1625	9,7869	13,6736	16,4575	
16	18,8275	10,2031	6,3164	3,5325	1,7167
17	18,9275	10,3031	6,4164	3,6325	1,8167
18	8,1675	0,4569	4,3436	7,1275	8,9433
19	21,5525	30,1769			
20	15,6125	24,2369			
Distancia Media Recorrida [m]	0,498275	0,4120	0,3732	0,3453	0,3272
Velocidad [m/s]	0,9135	0,7554	0,6841	0,6331	0,5998
Tiempo entre imágenes[s]	0,5455				
Desviación típica [cm]	46,0697	14,0269	10,1366	8,0818	4,9940
Dispersión [-]	92,46%	34,04%	27,16%	23,40%	15,26%

Figura 5.31: Resultados obtenidos en la situación nº4.

Tal y como puede verse en la imagen 5.31, aparecen señalados en rojo aquellos puntos que se han ido eliminando en cada iteración. Como vemos, se han realizado 4 iteraciones, parándose cuando los valores de las desviaciones se encontraban entre 0 y 9 cm. Además, se incluye el nuevo valor de distancia recorrida, velocidad, desviación típica y dispersión obtenido en cada iteración. Para obtener el resultado final vemos que se han empleado tan sólo nueve de los puntos que disponíamos inicialmente, siendo la distancia media recorrida de 0.3272 metros y la velocidad de 0.5998 m/s. En este caso, aunque finalmente sólo hayamos empleado nueve de los puntos, vamos a dar por bueno el resultado. Sin embargo, siempre será deseable disponer de una mayor representación de puntos, ya que la precisión del resultado será mayor.

CAPÍTULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

En este último capítulo de la memoria, se van a comentar tanto las conclusiones obtenidas a lo largo de la realización de este proyecto, como aquellas que se han extraído de los resultados conseguidos en las diferentes situaciones en las que se ha probado la herramienta desarrollada y que se muestran en el capítulo 5 de este documento.

En la primera parte del proyecto, se realizó un estudio de diversos detectores de puntos de interés para determinar cuál de ellos ofrecía un mejor rendimiento y se adaptaba mejor a las necesidades de este proyecto. A la vista de los resultados (se pueden ver en la sección 4.6), la mejor opción de las estudiadas es el detector SURF [2] implementado por Herbert Bay, Andreas Ess, Tinne Tuytelaars y Luc Van Gool. Este detector presentaba ciertas ventajas importantes respecto a otras posibilidades analizadas como son el detector SIFT (ver sección 4.4) o el detector de esquinas de Harris (ver sección 4.1). Algunas de estas ventajas son:

- Mayor rapidez de computación

- Facilidad de acceso al código
- Muy buena relación número de puntos detectados vs. errores en la etapa de emparejamiento
- Aporta gran cantidad de información

Por su parte, en la segunda etapa de este proyecto, se ha desarrollado la herramienta para obtener la distancia recorrida por el robot a partir de un par de imágenes estéreo. En primer lugar, el empleo de la cámara estéreo está justificado debido a su buen rendimiento, su precio y a la facilidad con la que permite reconstruir el escenario tridimensional visto por la cámara. Esto último es muy importante, ya que al contrario de lo que ocurría con las cámaras monoculares, con el sistema estéreo se puede obtener fácilmente el valor de la profundidad a partir de una serie de transformaciones geométricas y, además, no es necesario que el robot esté en movimiento para determinar la profundidad de los puntos vistos en la imagen.

En cuanto al algoritmo desarrollado, hay que decir que presenta un rendimiento más que aceptable, ya que permite obtener sin problemas tanto la distancia recorrida como la velocidad llevada por el robot en distintas situaciones y bajo diferentes condiciones de iluminación. Además, los errores cometidos en el cálculo, es decir las desviaciones producidas respecto al resultado final obtenido de cada uno de los puntos empleados como referencia, se mantienen dentro de un rango de $+/-10$ cm. en la mayoría de casos.

Por tanto, las principales conclusiones obtenidas en este proyecto son las siguientes:

- El uso de los detectores SURF ha permitido desarrollar una herramienta rápida que determina la distancia recorrida y la velocidad en aproximadamente 3 ó 4 segundos.
- El uso de la visión estereoscópica permite realizar una buena estimación de los valores de profundidad de los puntos capturados en las diferentes imágenes.
- Tras el ajuste estadístico de los resultados obtenidos, los errores cometidos en el cálculo de la distancia se mantienen teóricamente por debajo de los 10 cm. Aquí sería interesante comprobar los resultados teóricos

obtenidos con los valores reales a partir de una reconstrucción precisa del escenario en el que el robot desarrolla su actividad.

- La herramienta sólo permite obtener distancias recorridas relativamente pequeñas, ya que no es adecuado emplearla con puntos situados a profundidades superiores a los 10 metros, debido a que el número de errores cometidos y la magnitud de éstos aumenta (esto puede verse en la situación 2 descrita en el capítulo de resultados). Una posible solución para este problema sería emplear imágenes con resoluciones más grandes, ya que al disponer en la imagen de un mayor número de píxeles es posible obtener con mayor precisión las coordenadas de los puntos capturados por las imágenes. Sin embargo, esta solución tiene su contrapartida y es que los tiempos de cómputo se incrementan considerablemente.

- Tal y como se dice en el punto anterior, a mayores resoluciones de imagen menor será la magnitud de los errores cometidos.

- El uso de imágenes con poca textura también será una fuente de error a tener en cuenta, ya que los puntos localizados no tendrán tan bien determinadas sus coordenadas. Un ejemplo de esto sería las paredes de los pasillos del edificio Bethancourt (lugar donde se ha probado la herramienta).

- Aunque ya se ha realizado un tratamiento de los datos obtenidos como salida, sería interesante incluir en el algoritmo implementado una serie de cálculos estadísticos que nos permitiesen realizar un mejor ajuste de los resultados y descartar automáticamente los puntos con información incorrecta y que provocan errores en el valor de distancia obtenido.

Finalmente, es importante decir que a pesar del buen rendimiento obtenido con la herramienta desarrollada en este proyecto, ésta no supone una solución al problema de la navegación robótica ya que aún no permite realizar un cálculo de la distancia recorrida lo suficientemente preciso y fiable como para que pudiera dotar a un robot de cierta autonomía en su trayectoria. Sin embargo, esperamos que lo desarrollado en este proyecto pueda servir de ayuda a otras personas en futuros trabajos, de modo que algún día pueda alcanzarse el objetivo de crear un sistema de visión artificial lo suficientemente seguro y preciso como para que el robot pudiese prescindir del operador humano en su actividad.

APÉNDICE A

ESTIMACIÓN DEL MOVIMIENTO

Una vez tenemos nuestro sistema de visión descrito, debemos describir la manera en la que vamos a determinar el movimiento del robot. La primera consideración que debemos tener en cuenta en este punto, es que el movimiento realizado por el robot no es siempre en línea recta, sino que pueden producirse giros en su trayectoria. Por tanto, para no caer en errores a la hora de determinar la distancia recorrida debemos tener en cuenta la posibilidad de que el robot haya girado en su recorrido.

En primer lugar, para determinar el movimiento del robot, contamos con los datos que nos proporciona la secuencia de imágenes capturada con la cámara estéreo. De esta secuencia de imágenes, vamos a seleccionar dos de ellas pertenecientes a dos instantes diferentes y próximos en el tiempo. Tal y como hemos explicado anteriormente, con el sistema de visión estéreo podemos a partir de la imagen capturada en cada instante conocer de un punto concreto del espacio su posición respecto al robot, de modo que:

- En el instante t tendríamos las coordenadas "x" y "z" del punto bajo el sistema de referencia del robot (coincide con el sistema de referencia global).

- En el instante t+1 tendríamos las coordenadas "x" y "z" de ese mismo punto bajo el sistema de referencia del robot, pero éste estaría en esta ocasión girado un ángulo "θ" (que debido a que el movimiento de la cámara es solidario con el del robot, ese ángulo corresponde al giro realizado por éste) respecto al sistema de referencia global.

Considerado esto, se llega a una situación como la del dibujo:

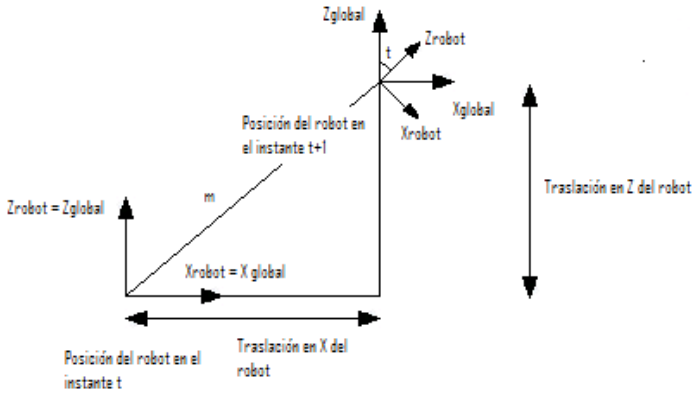


Figura A.1: Descripción movimiento del robot.

Donde "m" es la distancia recorrida por el robot y "t" es el giro realizado. Entonces, por Pitágoras podemos escribir lo siguiente:

$$m = \frac{Traslación\ en\ X}{sen(t)} \tag{A.1}$$

$$m = \frac{Traslación\ en\ Z}{cos(t)} \tag{A.2}$$

$$m^2 = (Traslación\ en\ X)^2 + (Traslación\ en\ Z)^2 \tag{A.3}$$

De este sistema, las incógnitas son "m" y el ángulo "t", ya que la traslación en X y la traslación en Z las conocemos:

$$Traslación\ en\ Z = Z_{t_1} cos(t) - X_{t_1} sen(t) - Z_{t_0} \tag{A.4}$$

$$Traslación\ en\ X = Z_{t_1} sen(t) + X_{t_1} cos(t) - X_{t_0} \tag{A.5}$$

Donde, $(Z_{t_1}cos(t) - X_{t_1}sen(t))$ y $(Z_{t_1}sen(t) + X_{t_1}cos(t))$ son las coordenadas del punto en el instante t+1 en el eje de coordenadas global, y "X_{t_0}" y "Z_{t_0}" son las coordenadas del punto en el instante t en el eje de coordenadas global. Tanto "Z_{t_1}" "X_{t_1}" como "Z_{t_0}" "X_{t_0}" son constantes conocidas.

Ahora, procedemos a resolver el sistema por el que es aparentemente el camino más sencillo:

$$\frac{Traslación\ en\ X}{sen(t)} = \frac{Traslación\ en\ Z}{cos(t)} \tag{A.6}$$

$$\frac{Traslación\ en\ X}{Traslación\ en\ Z} = tan(t) \tag{A.7}$$

Sustituyendo el valor de las traslaciones y despejando se llega a lo siguiente:

$$tan(t) = \frac{-2Z_{t_0}X_{t_0} \pm \sqrt{4X_{t_1}^2(Z_{t_0}^2 + X_{t_0}^2) - 4X_{t_1}^4}}{2(Z_{t_0}^2 - X_{t_1}^2)} \tag{A.8}$$

Con el arcotangente de la solución obtendríamos dos valores de "t". Eso tiene sentido ya que en función de si el robot avanza o retrocede el valor del ángulo será positivo o negativo. Por tanto, una vez calculado el ángulo de giro del robot, ya podemos sustituir en las expresiones A.4 y A.5 y obtener la traslación en X y la traslación en Z, con las que ya podremos obtener la distancia recorrida "m".

APÉNDICE B

CÁMARA POINT GREY BUMBLEBEE2 BB2-08S2C

A continuación, se van a describir las principales características de la cámara utilizada en el proyecto, la Point Grey Bumblebee2 BB2-08S2C (ver figura B.1).

En primer lugar, se trata de una cámara estéreo que consta de 2 sensores Sony CCD de 1/3" en color y blanco y negro. Captura imágenes de 1024x768 a 11 FPS (Frames por segundo) y la cámara se encuentra precalibrada para corregir internamente la distorsión de la lente, mostrando como salida la imagen corregida.

Figura B.1: Cámara Estéreo Bumblebee2

Las especificaciones técnicas de la cámara pueden verse en la figura B.2.

Especificación	Baja Resolución (640x480)	Alta Resolución (1024x768)
Sensor de imagen	Sony 1/3" Scan Progresivo CCD	
	ICX424 (648x488 max pixels)	ICX204 (1032x776 max pixels)
	7,4 mm pixels cuadrados	4,65 mm pixels cuadrados
Linea de base	12 cm	
Distancia focal	3,8 mm con 70° HFOV o 6mm con 50° HFOV	
Convertidor A/D	12 bit	
Salida de Datos de Video	Digital de 8, 16 y 24 bits	
Gamma	De 0.5 a 4	
Balance de blanco	Automático/Manual (Modelo de color)	
Frame Rates	48,30,15, 7.5, 3.75, 1.875 FPS	18, 7.5, 3.75, 1.875 FPS
Interfaz	6-pin IEEE-1394a para el contro y la transmisión de datos de video de la cámara. 4 pines de entrada/salida digital de propósito general	
Requirimientos de voltaje	8-32 V	
Consumo de energía	Menos de 2.5 W	
Ganancia	Automática/ Manual	
Shutter	Automáticao/ Manual	
	0.01 ms a 66.63 ms a 15 FPS	
Modos de Trigger	DCAM v1.31 Trigger Modes 0,1,3 y 14	
Ratio de señal-ruido	Mayor de 60 dB a una ganancia de 0dB	
Dimensiones	157x36x47.4 mm	
Peso	342 gramos	
Especificación de la cámara	IIDC 1394-based Digital Camera Specification v1.31	
Soporte de las lentes	2 x M12 soporte de microlentes	
Conformidad de emisiones	Conforme a las leyes de la CE y Part 15 Class A de FCC Rules	
Temperatura de operación	De 0° a 45°	
Temperatura de almacenamiento	De -30° a 60°	

Figura B.2: Especificaciones técnicas de la cámara Bumblebee2

Para un buen funcionamiento de la cámara, el fabricante recomienda la siguiente configuración del sistema:

- Windows XP Service Pack
- 512MB de memoria RAM
- Tarjeta de vídeo AGP (Puerto de gráficos acelerado) con 64MB de memoria de vídeo
- Bus 32-bit PCI estándar requerida para tarjeta IEEE-1394.
- MS Visual C++ 6.0 (Para compilar y ejecutar los programas de ejemplo)

Finalmente, en cuanto a las dimensiones de la cámara, éstas pueden apreciarse en la figura B.3.

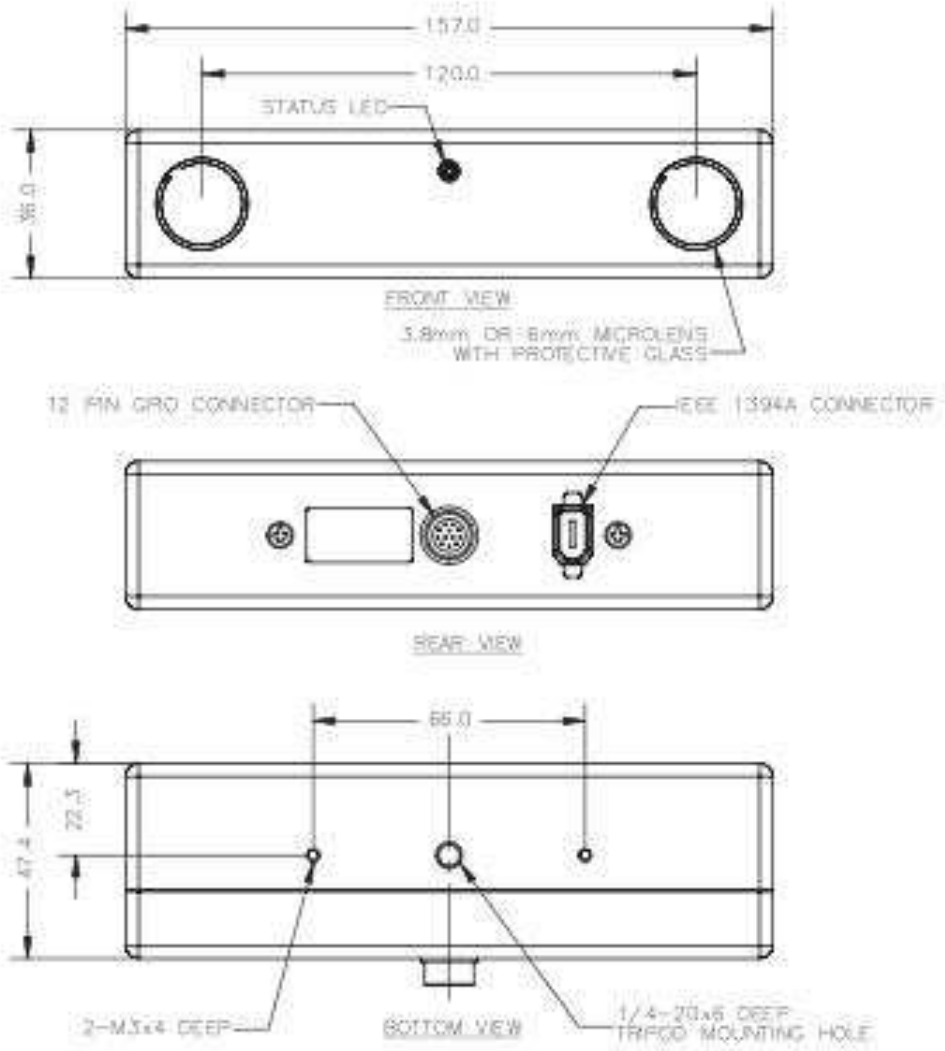


Figura B.3: Dimensiones de la cámara

BIBLIOGRAFÍA

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool: *"SURF: Speeded Up Robust Features"*. Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346–359, 2008.

[2] Página web principal de los descriptores SURF: *"http://www.vision.ee.ethz.ch/ surf/"*.

[3] David G. Lowe: *"Distinctive Image Features from Scale-Invariant Keypoints"*. International Journal of Computer Vision, pp. 91 - 110, 2003.

[4] David G. Lowe: *"Object recognition from local scale-invariant features"*. Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150-1157, 1999.

[5] Richard Hartley and Andrew Zisserman. *"Multiple View Geometry in computer vision"*. Cambridge University Press, 2003.

[6] Gilles Aubert and Pierre Kornprobst. *"Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations"*. Springer, 2006.

[7] Antonio Javier Gallego, Patricia Compañ, Pilar Arques, Carlos Villagrá, Rafael Molina: *"Detección de objetos y estimación de su profundidad mediante un algoritmo estéreo basado en segmentación"*. II Congreso Español de Informática (CEDI 07), 2007.

[8] Arturo de la Escalera. *"Visión por Computador: Fundamentos y Métodos"*. Prentice Hall, 2001.

[9] Guía de Usuario. *"Image Acquisition Toolbox: User's Guide"*. The Mathworks, 2008.

[10] Guía de Usuario. *"Image Processing Toolbox: User's Guide"*. The Mathworks, 2008.

[11] Guía de Usuario. *"Bumblebee2: Getting Started Manual"*. Point Grey Research, 2008.

[12] Federico Lecumberry: *"Cálculo de disparidad en imágenes estéreo, una comparación"*. III Workshop de Computación Gráfica, Imágenes y Visualización, 2005.

[13] Tesis Doctoral de Federico Lecumberry: *"Cálculo de Disparidad y Segmentación de Objetos en Secuencia de Video"*. Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, Agosto 2005.

[14] Mónica Ballesta, Arturo Gil, Óscar Martínez Mozos, Óscar Reinoso: *"Local Descriptors for Visual SLAM"*. Workshop on Robotics and Mathematics, Septiembre 2007.

[15] Mónica Ballesta, Arturo Gil, Óscar Martínez Mozos, Óscar Reinoso: *"Interest Point Detectors for Visual SLAM"*. Current Topics in Artificial Intelligence: 12th Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2007), pp. 170-179, 2007.

[16] Tesis Doctoral de Dirk Farin: *Automatic Video Segmentation Employing Object/Camera Modeling Techniques. "http://vca.ele.tue.nl/farin/phd/index.html"*. Technische Universiteit Eindhoven, Diciembre 2005.

[17] C. Harris y M. Stephens: *"A combined corner and edge detector"*. Proceedings of the 4th Alvey Vision Conference, pp. 147 - 151, 1988.

[18] Johannes Bauer, Niko Sünderhauf, Peter Protzel: *"Comparing Several Implementations of two Recently Published Feature Detectors"*. In Proc. of the International Conference on Intelligent and Autonomous Systems (IAV), September 2007.

[19] S. M. Smith and J.M Brady: *"SUSAN-A new approach to low level image processing"*. International Journal of Computer Vision, vol. 23, no. 34, pp. 45-78, 1997.

[20] Tinne Tuytelaars and Krystian Mikolajczyk: *"Local Invariant Feature Detectors: A Survey"*. Foundations and Trends in Computer Graphics and Vision, vol. 3, pp. 177-280, January 2008.

[21] Tony Lindeberg: *"Feature Detection with Automatic Scale Selection"*. International Journal of Computer Vision, vol.30, pp. 79-116, 1998.

[22] Krystian Mikolajczyk and Cordelia Schmid: *"Indexing based on scale invariant interest points"*. In Proc. Eighth IEEE International Conference on Computer Vision (ICCV), vol. 1, pp. 525-531, 2001.

[23] Krystian Mikolajczyk and Cordelia Schmid: *"An affine invariant interest point detector"*. 7th European Conference on Computer Vision, vol. 1, pp. 128-142, 2002.

[24] Herbert Bay, Beat Fasel and Luc Van Gool: *"Interactive Museum Guide: Fast and Robust Recognition of Museum Objects"*. First International Workshop on Mobile Vision (IMV 2006), May 2006.

[25] Luke Ledwich and Stefan Williams: *"Reduced SIFT Features For Image Retrieval and Indoor Localisation"*. In Australian Conference on Robotics and Automation (ACRA), 2004.

[26] A.C.Murillo, J.J.Guerrero and C.Sagüés: *"SURF features for efficient robot localization with omnidirectional images"*. IEEE International Conference on Robotics and Automation, Roma, Italy, 2007.

[27] Vikram Srivastava and Prashant Goyal: *"An Efficient Image Identification Algorithm using Scale Invariant Feature Detection"*. Stanford University, 2007.

[28] Behzad Shahraray and Michael K. Brown: *"Robust Depth Estimation from Optical Flow"*. Second International Conference on Computer Vision, pp. 641-650, 1988.

[29] M.Brown and D.G.Lowe: *"Invariant features from interest point groups"*. In British Machine Vision Conference, Cardiff, Wales, pp. 656-665, 2002.

[30] Stephen Se, David Lowe and Jim Little: *"Global Localization using Distinctive Visual Features"*. Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, vol. 1, pp. 226-231, October 2002.

[31] Sebastian Thrun, Dieter Fox, Wolfram Burgard and Frank Dellaert: *"Robust Monte Carlo localization for mobile robots"*. Artificial Intelligence, Vol. 128, pp. 99-141, May 2001.

[32] Jens-Steffen Gutmann, Wolfram Burgard, Dieter Fox and Kurt Konolige: *"An Experimental Comparison of Localization Methods"*. In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1998.

[33] Jens-Steffen Gutmann and Dieter Fox: *"An Experimental Comparison of Localization Methods Continued"*. IEEE/RSJ International Conference on Intelligent Robots and System, vol. 1, pp. 454-459, 2002.

[34] Arturo Gil, Óscar Reinoso, Asunción Vicente, César Fernández and Luis Payá: *"Monte Carlo Localization Using SIFT Features"*. IbPRIA 2005 : Iberian conference on pattern recognition and image analysis, pp. 623-630, 2005.

[35] Página web de la implementación SIFT de Andrea Vedaldi: *"http://www.vlfeat.org/ vedaldi/code/sift.html"*.

[36] Página web de la implementación SIFT de David Lowe: *"http://people.cs.ubc.ca/ lowe/keypoints/"*.

[37] Página web de la implementación SIFT de la Universidad de Stanford: *"http://robots.stanford.edu/cs223b04/project9.html"*.

[38] Página web de la implementación SIFT de Rob Hess (Oregon State University): *"http://web.engr.oregonstate.edu/ hess/"*.

[39] Página web de la implementación SIFT de Sebastian Nowozin (Universidad de Berlín): *"http://user.cs.tu-berlin.de/ nowozin/libsift/"*.